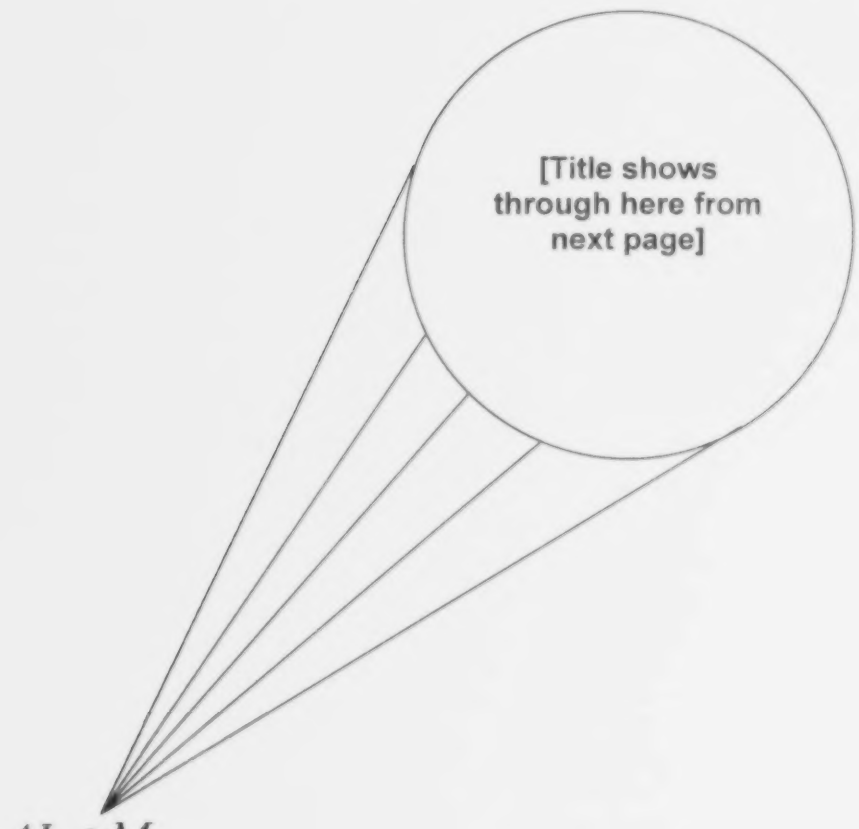
Title:	A Generic, Agent-Based Framework for the Design and Development of UAV/UCAV Control Systems
Author:	J. L. Edwards
Date:	27 February 2004
Document No.:	DRDC Toronto CR 2004-062
Contractor:	Artificial Intelligence Management and Development Corporation (AIMDC) 206 Keewatin Avenue, Toronto, Ontario M4P 1Z8, CANADA
Report No.:	AIMDC (AC261, February, 2004)

[Note: This page is not printed; Print page entitled: Defence R&D Cover Sheet]



AIM

Artificial Intelligence
Management and Development Corporation

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

© Her Majesty the Queen as represented by the Minister of National Defence, 2004
© Sa majesté la reine, représentée par le ministre de la Défense nationale, 2004

Author: J. L. Edwards, Artificial Intelligence Management and Development Corporation
Contractor: Artificial Intelligence Management and Development Corporation (AIMDC)
206 Keewatin Ave., Toronto, Ontario, Canada M4P-1Z8
Contract: W7711-037857/001/TOR
Contract Scientific Authority:
Dr. Ming Hou,
Human Factors Research and Engineering Section
Establishment: Defence R&D Canada Toronto (DRDC Toronto)
Document No.: DRDC TORONTO CR 2004-062
Date: 27 February 2004
Report No.: AIMDC (AC261, February, 2004)

Generic Agent-Based Framework For UAV/UCAV Systems

- Final Report -

Disclaimer: The scientific or technical validity of this Contract Report is entirely the responsibility of the contractor and the contents do not necessarily have the approval or endorsement of Defence R&D Canada.

© Her Majesty the Queen as represented by the Minister of National Defence, 2004
© Sa majesté la reine, représentée par le ministre de la Défense nationale, 2004

Abstract

Unmanned Air Vehicles (UAVs) and Unmanned Combat Air Vehicles (UCAVs) are being investigated for use as a new Integrated Intelligence, Surveillance, and Reconnaissance (IISR) platform within the Canadian Forces. At the moment UAV/UCAV control is operator intensive and can involve high levels of workload. In an effort to alleviate those conditions and reduce manning requirements, the current project examined a variety of theoretical approaches to construct a comprehensive, integrated approach to the design and implementation of an intelligent, adaptive, agent-based system for UAV/UCAV control. The resulting generic framework was constructed from the elements of the following design approaches: CommonKADS, MAS-CommonKADS, IDEF Standards, Explicit Models Design, Perceptual Control Theory and Ecological Interface Design. This report provides overviews of each of those approaches and highlights common and complementary elements as part of a recommended generic framework. A sequence for applying the generic framework is provided. The proposed integration of the above techniques into a comprehensive, cross-disciplinary design approach will help serve the goals of reducing operator workloads and manning requirements, while generating a robust, maintainable and reliable system.

Résumé

Les véhicules aériens télépilotés (UAV) et les drones de combat UCAV font maintenant l'objet d'une étude en vue de les utiliser comme une nouvelle plate-forme intégrée de renseignement, de surveillance et de reconnaissance (IISR) au sein des Forces canadiennes. À l'heure actuelle, le pilotage des UAV/UCAV repose intensément sur l'opérateur et peut donner lieu à de hauts niveaux de charge de travail. Afin d'améliorer ces conditions et réduire les besoins d'effectifs, le projet actuel a étudié diverses approches théoriques pour élaborer une méthode intégrée et globale visant à concevoir et à mettre en œuvre un système de commande intelligent, adaptatif et axé sur l'agent pour l'UAV/UCAV. Le cadre générique qui en découlait a été construit à partir des éléments des approches conceptuelles suivantes : CommonKADS, MAS-CommonKADS, normes IDEF, conception de modèles explicites, théorie du contrôle perceptuel (PCT) et conception d'interfaces écologiques. Le présent rapport donne un aperçu de chacune des approches et précise les éléments communs et les éléments complémentaires du cadre générique recommandé. Une séquence visant l'application de ce cadre est également fourni. L'intégration des techniques susmentionnées dans une approche conceptuelle interdisciplinaire complète contribuera à l'atteinte des buts visés, soit réduire la charge de travail des opérateurs et les besoins d'effectifs, tout en créant un système robuste, facile à maintenir et fiable.

This page intentionally left blank.

Executive Summary

Unmanned Air Vehicles (UAVs) and Unmanned Combat Air Vehicles (UCAVs) are being investigated for use as a new Integrated Intelligence, Surveillance, and Reconnaissance (IISR) platform within the Canadian Forces. At the moment UAV/UCAV control is operator intensive and can involve high levels of workload. In an effort to alleviate those conditions and to reduce manning requirements, the current project examined a variety of theoretical approaches to construct a comprehensive, integrated approach to the design and implementation of an intelligent, adaptive, agent-based system for UAV/UCAV control.

The resulting generic framework is composed of elements from the following design approaches:

- **CommonKADS (CK)** – a knowledge management and engineering methodology that guides the systematic analysis and design of intelligent systems;
- **MAS-CommonKADS (MAS-CK)** – a modification of the CommonKADS methodology that adds an infrastructure for multiple agent development;
- **IDEF Standards** – a complement to the CommonKADS methodology through its more effective support for temporal modelling and ontology construction;
- **Explicit Models Design (EMD)** – a methodology for building models that identify and compartmentalise the knowledge required by intelligent systems;
- **Perceptual Control Theory (PCT)** – a feedback control system model for goal-directed behaviour in a system;
- **Ecological Interface Design (EID)** – a set of techniques for constructing safe and reliable user interfaces based on levels of cognitive control.

Overviews of the individual approaches are given, and common and complementary elements of the framework are highlighted. The integration of the above techniques and methods into a comprehensive, cross-disciplinary design approach should serve the goals of reducing operator workloads and manning requirements while generating a robust, maintainable and reliable system.

Following is an overview of the recommended procedure for designing and implementing a knowledge system within the generic framework. Steps are designed to be pursued in the sequence presented. To facilitate an understanding of the procedure, a legend is provided of the models that comprise the CommonKADS knowledge and engineering methodology and those used in Explicit Models Design.

CommonKADS:

- Organisation Model;
- Task Model (CK);
- Agent Model;
- Knowledge Model;
- Communication Model;
- Co-ordination Model (MAS-CommonKADS);
- Design Model.

Explicit Models Design:

- Task Model (EMD);
- User Model;
- System Model;
- Dialogue Model;
- World Model.

The Proposed Generic Framework

- Construct the Organisation Model to describe the command and control structure within which the project will be developed;
- Construct the Task Model (CK), including task hierarchies for all agents identified above (use IDEF3 to represent the hierarchies);
- Construct the Agent Model identifying all user and system agents and their relationships;
- Adapt the Task Model (CK) to a five level Abstraction Hierarchy according to the levels specified by the EID approach;
- Generate the Task Model (EMD) by extending the Task Model (CK) to produce task hierarchies for all agents using PCT-based hierarchical goal analysis;
- Develop the User Model according to the need to track user preferences and knowledge;
- Specify the content of the System Model to enable representation and use of system preferences and knowledge;
- Design the World Model to contain required information about the environment necessary for the knowledge system to operate effectively;
- Specify the Dialogue Model, Communication Model and Co-ordination Model to govern the format and content of communication among agents (ensure that the ability exists for agents to provide feedback to one another);
- Use IDEF5 to design an ontology to represent the contents of all Explicit Models;
- Develop the Knowledge Model to encapsulate the ontology and an associated knowledge base containing information from all Explicit Models;
- Within the Knowledge Model, represent the Task Model (EMD) as a hierarchy of PCT loops that use plan recognition and plan generation to form input perceptions and output behaviours;
- Create the Design Model to produce design specifications for the target knowledge-based system;
- Apply EID techniques to develop specifications for the user interface to the knowledge system.

Edwards, J. L. 2004. A Generic, Agent-Based Framework for the Design and Development of UAV/UCAV Control Systems. AC261, DRDC Toronto CR 2004-062.

Sommaire

Les véhicules aériens télépilotés (UAV) et les drones de combat UCAV font maintenant l'objet d'une étude en vue de les utiliser comme une nouvelle plate-forme intégrée de renseignement, de surveillance et de reconnaissance (IISR) au sein des Forces canadiennes. À l'heure actuelle, le pilotage des UAV/UCAV repose intensément sur l'opérateur et peut donner lieu à de hauts niveaux de charge de travail. Afin d'améliorer ces conditions et réduire les besoins d'effectifs, le projet actuel a étudié diverses approches théoriques pour élaborer une méthode intégrée et globale visant à concevoir et à mettre en œuvre un système de commande intelligent, adaptatif et axé sur l'agent pour l'UAV/UCAV.

L'étude a abouti à un cadre générique comprend des éléments des approches conceptuelles suivantes :

- **CommonKADS (CK)** – méthodologie de gestion des connaissances et d'ingénierie, qui guide l'analyse systématique et la conception des systèmes intelligents;
- **MAS-CommonKADS (MAS-CK)** – modification de la méthodologie CommonKADS qui ajoute une infrastructure pour l'élaboration d'agent multiple;
- **Normes IDEF**– complémente la méthodologie CommonKADS par le fait qu'elles se prêtent plus efficacement à la modélisation temporelle et à la construction ontologique;
- **Conception de modèles explicites (EMD)** – méthodologie servant à construire des modèles qui déterminent et cloisonnent les connaissances requises par les systèmes intelligents;
- **Théorie du contrôle perceptuel (PCT)** – modèle de système de contrôle de rétroaction pour le comportement guidé par le but dans un système;
- **Conception d'interfaces écologiques (CIE)** – série de techniques servant à construire des interfaces-utilisateur sur la base des niveaux du contrôle cognitif.

Le présent rapport fournit un aperçu de chacune des approches et précise les éléments communs et les éléments complémentaires du cadre. L'intégration des techniques et méthodes susmentionnées dans une approche conceptuelle interdisciplinaire complète devrait contribuer à l'atteinte des buts visés, soit réduire la charge de travail des opérateurs et les besoins d'effectifs, tout en créant un système robuste, facile à maintenir et fiable.

Un aperçu de la procédure recommandée pour concevoir et mettre en œuvre le système des connaissances dans le cadre générique est donné immédiatement ci-dessous. Une légende des modèles comprenant la méthodologie de gestion des connaissances et d'ingénierie CommonKADS et les modèles qui utilisent la conception de modèles explicites est fournie afin de faciliter la compréhension de la procédure.

CommonKADS

- Modèle d'organisation;
- Modèle des tâches (CK);
- Modèle d'agent;
- Modèle de connaissances;
- Modèle de communication;
- Modèle de coordination (MAS-CommonKADS);
- Modèle de conception.

Conception de modèles explicites

- Modèle des tâches (EMD);
- Modèle de l'utilisateur;
- Modèle de système;
- Modèle de dialogue;
- Modèle du monde.

Le cadre générique proposé

- Construire le modèle d'organisation pour de décrire la structure de commande et de contrôle dans laquelle le projet se développera;
- Construire le modèle des tâches (CK), y compris la hiérarchie des tâches pour tous les agents susmentionnés (se servir de l'IDEF3 pour représenter les hiérarchies);
- Construire le modèle d'agent identifiant tous les agents utilisateur et tous les agents de système et leurs relations;
- Adapter le modèle des tâches (CK) à une hiérarchie d'abstraction à cinq niveaux pour tous les agents, à l'aide de l'analyse des buts axée sur la PCT;
- Créer le modèle des tâches (EMD) en élargissant le modèle des tâches (CK) afin d'engendrer des hiérarchies pour tous les agents grâce à l'analyse hiérarchique des buts axée sur la PCT;
- Mettre au point le modèle de l'utilisateur selon le besoin de cerner les préférences et les connaissances de l'utilisateur;
- Préciser le contenu du modèle de système afin de mettre en service la représentation et l'utilisation des préférences et des connaissances du système;
- Concevoir le modèle du monde pour contenir les renseignements requis sur l'environnement dont le système des connaissances a besoin pour fonctionner efficacement;
- Préciser le modèle de dialogue, le modèle de communication et le modèle de coordination pour régir le format et le contenu de la communication parmi les agents (s'assurer que les agents ont l'aptitude de fournir de la rétroaction entre eux);
- Se servir de l'IDEF5 pour concevoir une ontologie qui servira à représenter les contenus de tous les modèles explicites;
- Mettre au point le modèle des connaissances pour encapsuler l'ontologie et une base de connaissances associée contenant des renseignements émanant des modèles explicites;
- Dans le modèle des connaissances, représenter le modèle des tâches (EMD) comme une hiérarchie des boucles PCT qui utilisent la reconnaissance de plans et la génération de plans pour former des perceptions d'entrée et des comportements de sortie;
- Créer le modèle des connaissances pour déterminer les spécifications de conception relatives au système cible axé sur les connaissances;
- Appliquer les techniques EID servant à élaborer des spécifications pour l'interface-utilisateur au système des connaissances.

Table of Contents

Abstract i

Résumé i

Executive Summary iii

The Proposed Generic Framework iv

Sommaire v

Le cadre générique proposé vi

Table of Contents vii

List of Figures ix

List of tables x

Background 1

Generic Framework 2

The CommonKADS Methodology 3

Overview of CommonKADS 3

Application of CommonKADS to the UAV/UCAV System 4

Organisation Model 4

Task Model 5

Agent Model 6

Knowledge Model 6

Communication Model 8

Design Model 9

Incorporation of IDEF Elements 11

Overview of the IDEF Standards 11

IDEF's Contribution to the Framework 12

IDEF3: Process Modelling 12

IDEF5: Ontology Modelling 12

CommonKADS and IDEF Integration 14

Creation of Explicit Models 15

Overview of Explicit Models Design 15

EMD's Contributions to the Generic Framework 15

Task Model 16
System Model 16
User Model 17
World Model 18
Dialogue Model 18
Plan Recognition 18
Plan Generation 19
Feedback 19

The Role of Perceptual Control Theory 21
Overview of Perceptual Control Theory 21
Hierarchical Goal Analysis of Tasks For UAV/UCAV Control 23
Reorganisation and Machine-Learning 24
Using PCT Concepts in UAV/UCAV Control System Implementation 25
PCT Advantages in the UAV/UCAV Control System 28

Software Agent Paradigm 30
Overview of Agent-Oriented Development 30
Agents in CommonKADS 31
Agents and the IDEF Standards 32
Agents in Explicit Models Design 32
Hierarchical Goal Analysis (HGA) of Tasks in the Agent Network 33
UAV/UCAV Agent Hierarchy and PCT 33
The PACT Approach and Automation Levels 33

Human-Computer Interaction 35
Overview of Ecological Interface Design 35
Abstraction Hierarchy 36
Skills, Rules and Knowledge Taxonomy 37
EID Principles 38
Experimental Support for EID 39
Ecological Interface Design and the Generic Framework 39

Forms of Adaptive Aiding 41
Operator Training 41

Design and Implementation within the Generic Framework 43
Proposed Generic Framework 44

References 45

List of abbreviations/acronyms 49

List of Figures

Figure 1 The Perceptual Control Theory Model 22

Figure 2: Hierarchy of control loops. 26

List of tables

Table 1: PACT levels of autonomy 34

Background

One concept being pursued by the Canadian Forces is to introduce Unmanned Air Vehicles (UAVs) and/or Unmanned Combat Air Vehicles (UCAVs) as a new Integrated Intelligence, Surveillance, and Reconnaissance (IISR) platform (CFEC, 2002). The advantages offered by UAVs/UCAVs to the military commander are numerous, most notably in mission areas commonly categorised as "the dull, the dirty, and the dangerous." For example, due to its vantage point, one unmanned sentry equipped with automated cueing algorithms and multiple sensors could survey the same area as ten (or more) human sentries ("the dull"). UAVs could reconnoitre areas contaminated with radiological, chemical or biological agents without risk to human life ("the dirty"). UCAVs could perform the high-risk suppression of enemy air defences (SEAD) missions currently flown by manned EA-6s or F-16s ("the dangerous") with less need for supporting aircraft (UAVR, 2002). In such a role, UAVs would be perfect force multipliers, releasing manned aircraft for other roles. At the moment UAV/UCAV control is operator intensive and can involve high levels of workload.

UAVs/UCAVs will require operators to interact with automated systems that may include intelligent agents/interfaces, similar to users interacting with software assistants as they navigate through new software, e.g., as occurs in Microsoft Word. The intelligent/adaptive interfaces that we envision for UAV/UCAV operators are more sophisticated and will require knowledge about the mission goals, the operator's goals and states, as well as environmental states. In some cases, the agent will perform a function automatically, ask permission to act, provide information, ask for clarification or relinquish control to the operator.

Both the USAF and UK have emerging programs in Intelligent/Adaptive Interfaces (IAIs) for decision support technologies. IAIs are intended to manage information dynamically and provide the right data and information at the right time to support effective decision making. Such technologies also will be integrated into many, if not all, Canadian Forces' systems of the future. Defence Research & Development Canada (DRDC) has initiated a project for the development of IAI for advanced UAV/UCAV control under Thrust 13i.

The aim of this project is to develop, demonstrate and prioritise enabling technologies that can be applied to an operator interface that will support reduced manning and enhanced performance in complex military systems. This project will also produce a document that will provide preliminary design guidelines for IAIs.

The domain of application for the demonstration part of the project is advanced UAV/UCAV control, focusing on those technologies that increase the ratio of platforms to operator. The project involves the following phases and will last 3 years:

Year 1 Concept development and theoretical frameworks;

Year 2 Interface design and implementation; creation of the simulation environment;

Year 3 Demonstration of the concept and experimental evaluation; draft design guidelines for IAIs; generation of a follow-on Technology Demonstration proposal.

Currently, there are no established design guidelines for advanced adaptive interfaces and the goal of the current contract is to provide support for the overall project by helping to establish preliminary design guidelines for such interfaces, where one operator controls multiple UAVs and UCAVs. This will be accomplished in two ways, first, be reviewing frameworks for intelligent and adaptive interfaces and, second, by providing recommendations for a framework to investigate operator-

UAV/UCAV(-agent) interaction in Adaptive/Intelligent Interfaces. Establishing and refining that framework is central to dealing with interaction issues like stability and optimisation for dual and multiple agents.

Generic Framework

The current document examines a number of theoretical approaches that are suitable for creating an intelligent, adaptive interface for UAV/UCAV control. Those theoretical elements have been combined to produce a comprehensive, cross-disciplinary approach to intelligent system design and implementation: a **generic framework**. Although it has been developed for the specialised task of UAV/UCAV control, it is *generic* in that it has been designed to be applicable to many other complex military systems with high operator workloads.

Current UAV/UCAV control systems are associated with high operator workload and commonly require two operators for each remote aircraft. The proposed framework uses an agent-based approach in order to alleviate some of those demands upon users performing highly complex tasks in difficult mission scenarios. Intelligent software agents are well suited to taking over some responsibilities for users, allowing human operators to focus exclusively on tasks that require human oversight.

In the same way, software agents can also contribute to the goal of reducing manning requirements, permitting a single operator to control multiple aircraft. The distributed hardware needs of such systems lend themselves to an agent-based approach because the communication abilities of agents enable them to function in a network of machines. Finally, autonomous software agents can lead to a more robust system, particularly where the possibility exists of communication breakdown between operator and aircraft.

The approach to developing an intelligent, adaptive interface is guided by the use of a knowledge design and engineering methodology that combines elements of the CommonKADS and IDEF methods, Explicit Models Design and Perceptual Control Theory. Principles from the fields of Human-Computer Interaction and agent-oriented software development have also been incorporated.

The following sections present overviews of those individual components and describe how each contributes to the generic framework. Rationales are provided for their inclusion in the integrated approach. Because of all the interrelationships among elements of the constituent components of the framework, the order in which they appear in the following sections does not necessarily correspond to the order in which they should be designed and implemented. Consequently, the recommended sequence for applying the framework is presented near the end of the document in a section entitled, "Design and Implementation within the Generic Framework."

The CommonKADS Methodology

Overview of CommonKADS

"Intelligent" software, such as that required for the UAV/UCAV adaptive interface, is built using what are called **knowledge-based systems** (KBS), which imitate human reasoning. Such systems are typically composed of formal representations of knowledge about a particular domain and a mechanism that enables the system to "reason" about that knowledge through the application of inferences. This overview section examines the CommonKADS approach (Schreiber, Akkermans, Anjewierden, de Hoog, Shadbolt, Van de Velde & Wielinga, 2000) to analysing and designing *knowledge-based systems* and the role of that approach in the generic framework.

The CommonKADS methodology provides a framework designed specifically for developing knowledge systems. CommonKADS has several strengths that make it a suitable choice for use in developing an intelligent, adaptive interface for UAV/UCAV control. A key advantage is that it guides developers through a systematic process intended to settle all issues related to the design of the software before any code has been written. Such a process substantially reduces the likelihood that developers will have to revise the design once implementation is underway. Other benefits of the approach include improved maintainability, reliability and reusability of software components over conventional less comprehensive and *ad hoc* approaches, which are not guided by any systematic or theoretical criteria such as correctness, completeness and consistency. Those benefits will be discussed in greater detail in subsequent sections, but first an overview of the CommonKADS methodology is in order.

CommonKADS and its antecedent, the KADS (Knowledge Acquisition and Design Structuring) methodology, are the products of a collaboration of European researchers in the 1980's and 1990's and funded as a series of projects under the ESPRIT programme. The methodology provides a systematic approach to analysing organisational needs, identifying opportunities for the deployment of knowledge systems and ultimately designing an implementation of the requisite system.

The CommonKADS methodology provides a step-by-step approach to analysing a given problem domain. Questions are posed at each stage through standard "worksheets," which provide a systematic tool for documenting both questions and answers concerning key issues. Individual, numbered guidelines are also offered to assist with the construction of more detailed components of the models.

An important term often used to refer to domains that could benefit from intelligent system design is **knowledge-intensive.** More specifically, the term is used to describe tasks that would benefit from an implementation that features a formalised representation of knowledge and an associated inference mechanism. Many, if not most, tasks carried out in software design do not require such treatment, and can be addressed with traditional software implementation approaches. For example, most help systems, even for highly complex software, typically would not benefit from a knowledge-based approach since they simply involve displaying simple help material when the user requests it. On the other hand, a *knowledge-intensive* help system with inference capabilities would benefit from methodologies such as CommonKADS because of the tools they provide for knowledge analysis, design and implementation that cannot be found in standard programming approaches. Examples of *knowledge-intensive* tasks include those that mimic human activities, such as diagnosis, planning and design.

UAV/UCAV control is *knowledge-intensive* in the context of the proposed adaptive interface because the system requires the ability to receive information about the aircraft and its environment, reason about that information in the context of its knowledge of the outside world and the operator's current needs, and decide on the most effective way to provide assistance. Therefore, a knowledge-based system approach has been adopted for the generic framework.

Other methodologies exist for use in developing knowledge systems, such as Methontology, TOVE and IDEF5, but they are concerned primarily with ontology specification and lack CommonKADS' support for producing detailed design specifications for an entire knowledge system. (See Jones, Bench-Capon and Visser (1998) for a discussion of those other techniques.)

Application of CommonKADS to the UAV/UCAV System

Since CommonKADS was developed specifically for guiding the analysis, design and implementation of knowledge-based systems, it is very well suited to producing such a system for UAV/UCAV control. As the first component of the generic framework, it provides a solid foundation on which to build.

The process of applying CommonKADS to the UAV/UCAV interface will involve the specification of the following six models:

- Organisation Model;
- Task Model;
- Agent Model;
- Knowledge Model;
- Communication Model;
- Design Model.

Organisation Model

The Organisation Model is assembled during the initial feasibility and assessment phase of a CommonKADS analysis. The primary emphasis of that phase is to examine organisational or business processes that could benefit from the implementation of a knowledge system. Coupled with that is the subsequent feasibility analysis that weighs the costs associated with such an implementation against the projected benefits. Thus, a system must be sufficiently *knowledge-intensive* to warrant its implementation using CommonKADS. Worksheet questions help to identify the structure of the organisation, relevant processes, people and resources involved and knowledge assets required.

Organisationally, it has already been determined that the addition of a knowledge-based system for UAV/UCAV control would be beneficial and feasible. That decision has been taken because conventional systems have proven inadequate to achieve the goal of reduced manning and workload requirements. As a consequence, it is unlikely that an organisational analysis, if conducted, will figure as prominently as it would in other projects.

However, it should be noted that the authors of the CommonKADS approach emphasise the importance of an initial phase in which a systematic analysis of the organisation is conducted. They point to situations where applying the methodology has revealed needs for a knowledge system that are substantially different from those projected at the outset of analysis. The use of CommonKADS to design an adaptive interface for UAV/UCAV control likely would not feature a detailed examination of organisational issues, however, an abbreviated form of organisational analysis of the command and control structure surrounding UAV/UCAV deployment could offer insights into the context in which the knowledge system is to be used.

Furthermore, an organisational analysis in the UAV/UCAV context could reveal a need to involve in particular ways not only the operators, but also those higher in the chain of command. The resulting knowledge system might specify useful ways in which real-time consultation could occur among operators and their commanders while UAV/UCAV missions were in progress. Such a system could apply rules to extract critical information and convey it to those in command, enhancing collaboration among decision-makers and facilitating mission re-planning during operations.

Task Model

The Task Model in CommonKADS is created primarily as part of the organisational analysis and feasibility assessment and focuses on high-level tasks and goals of agents in the system. Tasks are examined with a view to identifying those that are sufficiently *knowledge-intensive* that they would benefit from the implementation of a knowledge-based system.

To create an explicit representation, tasks and sub-tasks are represented using flow diagrams from the Unified Modelling Language (UML). UML is the standard technique in CommonKADS for schematically depicting activity, state and class diagrams that can represent a wide range of concepts such as data flow, inferences and task structures.

Construction of the Task Model involves the creation of a task hierarchy, in which tasks are decomposed into sub-tasks. Questions posed during Task Model creation relate to constituent sub-tasks, the agents and objects involved, the timing of the task and knowledge required for performing it. That information can then be referred to later during knowledge model construction.

The CommonKADS approach does not specify the extent to which tasks are to be decomposed into sub-tasks, as that is project-dependent. For a given project, practical boundaries must be established during the design phase to minimise the collection of irrelevant material. For example, in the UAV/UCAV domain, it is certainly relevant to examine tasks performed by aircraft operators and their direct superiors. There is also an argument for identifying tasks performed at higher levels in the chain of command that have direct bearing on UAV/UCAV missions. However, it would not be productive to examine all tasks at that level, since many would have no relation to the target knowledge system.

Similarly, there must be a boundary on the depth of sub-task decomposition. For example, the UAV/UCAV task of landing the aircraft could be composed of a very large number of sub-tasks surrounding the manipulation of individual controls. During the Task Model phase of the CommonKADS analysis, it is not necessary to address that level of detail since the intention is to develop general specifications for agent responsibilities and knowledge requirements.

In contrast, the Task Model in Explicit Models Design allows developers to look beyond those high-level goals to get the full picture of tasks at all levels. For more detail, refer to the "Creation of Explicit Models" section.

In the UAV/UCAV domain, the focus of the CommonKADS Task Model construction would be the high-level mission tasks of the operator. That would include elements such as: mission planning, UAV/UCAV launch, transiting to the area of operations, performing reconnaissance or SEAD, returning to base, landing and post-mission debriefing.

Although CommonKADS specifies that UML be used for representing task hierarchies, there are limitations to what that language currently is able to express and, consequently, it is recommended that an alternative complement to UML be used in the UAV/UCAV work. Those limitations relate to the representation of temporal constraints, such as concurrency of tasks and their performance in real-time. As those are likely to be crucial aspects for modelling in the air domain, it is recommended that the IDEF3 language be used for graphically representing tasks and sub-tasks in the UAV/UCAV setting (more detail is provided in the "IDEF3" section below).

In contrast to the Organisation Model, the Task Model has greater potential for providing insights into how to proceed with design of the UAV/UCAV control system. A formal statement of tasks and sub-tasks will help to identify responsibilities that can be carried out by software agents. Such tasks include:

- monitoring the operator's activities;
- inferring the user's immediate goals and broader plans;
- generating system plans to assist the operator in the most effective way given current circumstances.

Identification of those tasks should be performed in consultation with subject-matter experts.

It should be noted that the creation of a task hierarchy is fundamental to several of the approaches that comprise the generic framework, including Explicit Models Design, Perceptual Control Theory and Ecological Interface Design. Task hierarchies are discussed further in those sections that follow.

Agent Model

Like the two models already described, the Agent Model is developed during the initial feasibility phase. It is used to identify the participants in the itemised tasks so that their responsibilities can be incorporated into any resulting knowledge system. That process also assists with identifying expert sources of knowledge that can be useful in supplying information and providing rules for the knowledge base.

Construction of the Agent Model involves examining the tasks in which each agent is involved, the other agents with which it communicates and knowledge that is required to complete its tasks.

Software agents will be of critical importance in achieving the goal of reduced manning and operator workload in the UAV/UCAV domain. Agents are discussed in greater detail in the "Software Agent Paradigm" section below.

Knowledge Model

The Knowledge Model contains a detailed enumeration of all knowledge required to perform the tasks that the system will be performing. Thus, most of the architecture of the knowledge system is designed during the formulation of that model. The Knowledge Model is a specification of

requirements that is independent of the implementation approach. Thus, when a "rule" is specified, it is not a rule as it would be entered into a particular knowledge-based system, but it is rather a rule as it might be characterised by a human expert, without concern for adapting it to the needs of a particular implementation environment, thereby allowing an extra level of freedom for the designer.

The Knowledge Model is subdivided into three categories: "domain," "inference" and "task" knowledge. **Domain knowledge** contains all of the data used by the application, which, in object-oriented terminology, would correspond to the class definitions and object instances. CommonKADS does not follow the object-oriented paradigm, however, using the term "concept" instead of "class." Like classes, concepts have attributes and those attributes can be assigned values. Unlike classes, however, concepts do not have encapsulated methods but, instead, the methods are contained in the other two components of the knowledge model. Concepts can have subtype and supertype relationships to other concepts, thereby enabling the traditional object-oriented notions of inheritance and subsumption.

In the case of the adaptive interface for UAV/UCAV control, the domain knowledge would include knowledge relating to the capabilities and functions of the aircraft, including operator and system tasks described in the Task Model. Knowledge about the environment in which the aircraft operates also would be contained in the domain knowledge.

The second component of the Knowledge Model is the **inference knowledge**, which is a collection of methods that act on the domain knowledge. Thus, they are not exclusively functions that make inferences in the traditional knowledge system sense, but can include any method that acts on domain knowledge. Methods are separated from objects in CommonKADS to allow them to be individually reusable.

CommonKADS provides a catalogue of inference templates, an approach that has multiple advantages. Creating methods that can be applied generally allows them to be reused readily in other applications. Because they have been applied in other situations, they come pre-tested and therefore contribute to the overall reliability of the knowledge system. Standard inferences include: classify, compare, evaluate and predict.

Examples of inference knowledge in the UAV/UCAV control system would include methods for inferring goals from actions or hypothesising a plan based on a user's actions.

The final component of the Knowledge Model is **task knowledge**, comprising a set of higher-level methods that implement a hierarchy of tasks and sub-tasks. At the lowest level, the sub-tasks make use of methods in the underlying inference knowledge layer, which in turn operate on the domain knowledge.

Task knowledge in the UAV/UCAV context would include a representation of the full task hierarchy derived in the specification of the EMD Task Model (see the "Explicit Models Design" section).

As with the inference knowledge, there are templates available for task knowledge and they share the same benefits of reusability and reliability. Templates that are provided include those for planning, scheduling and monitoring, all of which would be useful in the UAV/UCAV adaptive interface. Planning and scheduling are tasks required for **plan generation** and monitoring is performed in association with **plan recognition** (see the section on Explicit Models, below, for more detail on plan generation and recognition).

Knowledge modelling in any context typically involves the creation of an ontology, and CommonKADS is no exception. Ontologies are formally specified frameworks within which knowledge can be represented. A chief goal in producing an ontology is to identify patterns in the

knowledge and exploit those to produce a highly organised and concise specification. One of the main motivations for generating an ontology for a particular domain is to allow its reuse in other applications.

In CommonKADS, an ontology is referred to as a domain schema, which is a formal statement of the structure of all concepts, relations and rules for a particular domain. Maintained separately from the domain schema is the knowledge base, which is a database of specific instances of the concepts, relations and rules. That separation mirrors the distinction in object-oriented analysis between class definitions and instantiated objects.

CommonKADS does not provide much guidance for ontology construction and lacks concrete examples to illustrate how ontologies should be structured. For that reason, IDEF ontology description methods are recommended within the generic framework. Those methods are described in the "IDEF5" section below.

In order to describe fully the Knowledge Model for the UAV/UCAV system, it will be necessary to design the content of the various Explicit Models first. The section on "Explicit Models Design" describes how those models are constructed.

Communication Model

The purpose of the communication model is to describe communication that must occur among agents in the knowledge system. That can include dialogue that is both between the user and system agents, and between individual software agents.

Communication is broken down using a transaction model. For each pair of agents that must interact, a communication plan is constructed (often represented using UML) that outlines the flow of information and decisions affecting that flow. That is further decomposed into a detailed itemisation of individual transactions, where each one represents a message sent from one agent to another. Each transaction is described in terms of the agents involved, the content of the message and knowledge objects exchanged.

Standard patterns of communication are described in CommonKADS, such as the straightforward "Ask" and the associated "Reply," or slightly more complex exchanges such as, "Require," which can have "Agree" or "Reject" as responses. A library is offered for those and other standard modes of communication.

One type of user-system interaction in the UAV/UCAV system will involve the presentation of information by the system and possible acknowledgement from the user. (In situations where it would be disruptive to the user to provide explicit feedback, the system will infer through indirect means that the user has received the information.) In addition to that system-initiated communication, users will also be able to request information from the system, which necessitates a second form of dialogue. Interaction could also consist of a clarification dialogue between user and system agents whereby the system would seek information when the user's current intentions are ambiguous. Given the critical nature of some aspects of missions, care must be taken in all circumstances to avoid unnecessary requests for communication with the user. Guidelines specifying the nature of such communication could be a useful by-product of the Communication Model.

The Communication Model also describes dialogue that occurs among system agents, which will be critical in the UAV/UCAV adaptive interface to maintain co-ordination among semi-autonomous entities. Agent interaction in the Communication Model is revisited in the "Software Agent Paradigm" section below.

Design Model

The Design Model examines hardware and software issues related to the construction of the knowledge system. The aim is to take the implementation-independent specifications from the Knowledge and Communication Models and develop a detailed design for constructing the software application, and in the process preserve the structure of those models. That preservation of structure is achieved by coding the software so that the formal specification of the Knowledge Model can be read in directly to the knowledge system. That approach simplifies future refinement and maintenance of the models allowing changes to the formal specification of the knowledge model to be accommodated without the need for re-coding. Further, it facilitates reuse of the components within the models by isolating them from implementation details.

The first step in the design process is the specification of the system architecture, which is largely predefined in the CommonKADS methodology. Sommerville (1995) recommends that system architecture descriptions include three components:

- decomposition of the system into subsystems;
- overall control regimen;
- decomposition of subsystems into software modules.

Those components are specified in CommonKADS through the use of the Model-View-Controller (MVC) paradigm, originally developed for use in SmallTalk-80, the original object-oriented programming language. In that approach, the Application Model contains the rules, inference functions, and knowledge bases that are responsible for the main functionality of the application. The Views subsystem provides external views of the data in the application model, which can be in the form of a user-interface or can also involve the presentation of information to an external software system. The Controller handles the processing of events, the triggering of tasks and inferences, and the responsibilities of the Communication Model.

Although the MVC model was originally developed for use in an object-oriented environment (SmallTalk), it does not rely on properties of that paradigm and is therefore equally suitable for use with variants of object-oriented approaches, such as CommonKADS.

The next design step is identifying the target software and hardware platforms. It is recommended that CommonKADS systems be implemented in an object-oriented (O-O) environment, despite the fact that the methodology does not adhere to the O-O approach of encapsulating methods within objects. The required separation between objects and methods can be achieved within an O-O framework by creating separate objects for each. The O-O approach is recommended because most modern environments adhere to that paradigm and most developers are familiar with it. Also, necessary capabilities such as message-passing and multi-threading are typically supported in O-O systems.

Some suggested languages for implementing CommonKADS systems are Prolog and Java, but that is not to the exclusion of other possible environments. Examples of Prolog implementations are offered in Schreiber et al. (2000) and source code for those examples is provided on the CommonKADS web site at www.commonkads.uva.nl.

Prolog has the advantages of extensive inference capabilities and support for the declarative representation of knowledge. A key drawback to that language is its weak support for the object typing that would normally be available in a fully object-oriented environment. That can be

addressed with additional coding, and the example on the CommonKADS web site demonstrates how that can be accomplished in Prolog.

Java has full O-O typing facilities and also wide platform support. With the addition of the Java Expert System Shell (JESS), a full inference engine is also available, as is support for declarative knowledge specification. Lacking are examples and source code for CommonKADS implementations in Java, but such code could be modelled on available Prolog examples. If a language other than Prolog is used in the UAV/UCAV system, some work will be required to adapt the available CommonKADS code.

Once an implementation environment has been selected, the final step in constructing the Design Model is to create a detailed plan for implementation of the Application Model, Views and Controller, as well as the tasks, inferences and domain knowledge within the Knowledge Model. Many details of the plan are dependent on the chosen environment.

CommonKADS also includes guidelines on project management that are designed to accommodate the unique needs associated with knowledge projects. The development of a project is divided into phases, each encompassing the construction of one of the major CommonKADS models. A cyclical four-step process is recommended, where each repetition produces a more refined version than the last. The first step in the cycle involves establishing objectives for the current phase. The next step seeks to assess risks in the project, such as a lack of available expert consultants, which could affect the quality of the knowledge system. The third step is to produce a detailed plan for the current phase taking into account identified objectives and risks. Last, the development work is monitored and evaluated by the project manager and the results of that stage are fed into the beginning of the next phase in the cycle.

Incorporation of IDEF Elements

Overview of the IDEF Standards

The IDEF standards, like the CommonKADS methodology, provide a set of guidelines for system analysis and design. They are the product of a U.S. Air Force project called Integrated Computer-Aided Manufacturing (ICAM), which was instituted to look at methods for analysing and improving manufacturing operations. Out of that effort came the development of the ICAM Definition (IDEF) methodologies for analysing processes, activities and information needs within organisations. Manufacturing involves many activities that are common to a wide range of problem domains, such as enterprise management, process modelling, data storage and software design. Because the IDEF standards address those issues individually and without specific reference to manufacturing, they are sufficiently general as to be applicable to a broad range of problem domains..

IDEF and CommonKADS share a number of common aims, but there are also numerous differences in the two approaches. One important difference to note is that CommonKADS was developed exclusively as a methodology for developing knowledge-based systems (KBS), whereas the IDEF standards contain components that can be applied to a broader range of problem domains. As a consequence of the IDEF standards' generality, they lack some of the KBS-specific elements that CommonKADS provides, particularly with regard to knowledge system implementation.

The IDEF documents are a collection of numbered standards, each of which provides formal guidelines for analysis and design in a particular area. Although there were plans for standards from IDEF0 to IDEF14, many of those projects were never implemented in final form. Standards that were finalised are:

- IDEF0 (National Institute of Standards and Technology, 1993a), which governs function modelling;
- IDEF1 (Mayer, 1992), which describes information modelling;
- IDEF1X (National Institute of Standards and Technology, 1993a), which covers relational database modelling;
- IDEF3 (Mayer, Menzel, Painter, deWitte, Blinn & Perakath, 1995), which concerns process modelling;
- IDEF4 (Knowledge Based Systems, Inc., 1995), which governs object-oriented design;
- IDEF5 (Knowledge Based Systems, Inc., 1994), which describes ontology modelling.

The two standards relevant to the generic framework are IDEF3 and IDEF5.

IDEF's Contribution to the Framework

IDEF3: Process Modelling[1]

The CommonKADS approach uses UML for the schematic representation of processes and associated data, agents, tasks and inferences. At various stages in a CommonKADS analysis, the developer is asked to describe the processes carried out by the system being modelled. Typically, the clearest way to present such a description is through the use of a graphical modelling language. UML is one such language that is in widespread use, particularly in the area of software design, and that has led the authors of the CommonKADS approach to adopt UML for use within the methodology.

One of the drawbacks of UML is that it is inflexible in representing temporal relationships and constraints among those elements. Two important temporal concepts are synchronisation and real-time systems. In the case of the former, a distinction is made between synchronous and asynchronous activities, i.e., those that occur at the same time contrasted with those that do not. That distinction can have important consequences for the system being modelled, where, for example, tasks that can be carried out synchronously can shorten the total time required to complete a procedure and thereby increase the efficiency of the system. When modelling real-time systems, much more attention must be paid to the precise times at which events occur, not simply their relative occurrence, and that can have direct bearing on whether or not the modelled system will perform as intended.

Unlike UML, IDEF3 permits flexible modelling of temporal concepts. For example, symbolic representations exist for depicting whether multiple activities are synchronous or asynchronous and whether all activities must be complete before the next steps in the process can continue.

Modelling of real-time systems is possible using IDEF3's elaboration language, which allows symbolic representations of complex constraints that cannot be depicted using the schematic language alone. The language, based on a subset of the Knowledge Interchange Format, permits formal logical representations of process constraints and allows precise specification of event timings and durations.

There are a number of options for achieving increased flexibility for temporal constraint modelling in UML. The upcoming release of the language (UML 2.0) promises to offer more of such capabilities, and the addition of the Object Constraint Language (OCL) to UML also features added support for temporal modelling. OCL, developed at IBM in the late 1990s, permits the specification of pre-conditions and post-conditions in a UML model and those can be used to describe temporal constraints.

Because IDEF3 has full flexibility for temporal modelling, and because it has been in use for a long time, IDEF3 is recommended for process modelling in the UAV/UCAV generic framework.

IDEF5: Ontology Modelling

Another IDEF standard to be used in the generic framework is IDEF5, which, like CommonKADS, provides specifications for ontology modelling. CommonKADS provides techniques for developing ontologies, including guidance on expert knowledge elicitation, formal descriptions of concepts,

---

[1] Performance modelling requires the construction of detailed process models and that can be accomplished using IDEF3 methods, although IDEF3 lacks specific techniques for analysing performance. In the generic framework, human factors considerations associated with performance may be accommodated within the User Model in Explicit Models Design (EMD).

attributes and relations, and a formal language for representing ontologies. Although those are powerful components, ontology construction is explored in greater depth in IDEF5, with more guidance offered and more examples provided.

The IDEF5 approach has five steps:

- organising and scoping;
- data collection;
- data analysis;
- initial ontology development;
- ontology refinement and validation.

The organising and scoping phase examines the context and purpose of the project, and can make use of the material assembled during the specification of the Organisation, Task and Agent Models of CommonKADS. Data collection involves acquiring raw data by, for example, examining existing systems or eliciting knowledge from experts. Data analysis attempts to refine the raw data into a form more usable in an ontology. Initial ontology development creates a draft of the ontology which is further developed in the refinement and validation phase. The methodology divides each of those five stages into sub-steps and provides detailed guidelines on how to accomplish each.

IDEF5 includes two different languages for representing ontologies: one graphical and the other textual. The former, known as the **IDEF5 Schematic Language**, allows users to model concepts and relations using straightforward box and line diagrams. That permits the clearest views of the relationships in the knowledge for both the modeller and others.

Since there are limits to the amount of detail that can be incorporated conveniently into a diagrammatic representation, a second ontology language is provided: the **IDEF5 Elaboration Language**. That language is a textual representation, based on the Knowledge Interchange Format, that provides full support for representing complex concepts and relations in an ontology. The Elaboration Language supports declarative statements of knowledge and allows the use of standard logical, arithmetic and set theoretic operators, as in the following example:

(forall ?x (=> (and (human ?x) (>= (age-of ?x) 21)) (adult ?x))),

which specifies that all humans whose age is at least 21 are adults.

In contrast to its formal representation of declarative knowledge, IDEF5 does lack numerous components that CommonKADS offers for the implementation of a knowledge system. While it provides powerful tools for ontology creation, IDEF5 does not offer support for designing knowledge bases or for performing inference on the knowledge. Also, it does not deal with specific implementation issues, such as the design of software modules and recommendations on platform requirements. Fortunately, CommonKADS fills those gaps, and integrating the two techniques in the UAV/UCAV generic framework would leverage the strengths of both approaches.

CommonKADS and IDEF Integration

The integration of CommonKADS and IDEF techniques leads to a robust framework for developing knowledge-based systems. While the discussion has been focussed on their applicability in the UAV/UCAV domain, they are equally suitable for any knowledge-intensive task. For example, a combined CommonKADS/IDEF approach is being used to design a generalisable help system that is being developed in the context of the *LOCATE* workspace design tool (Edwards, Scott & Hendy, 2003).

Furthermore, other IDEF standards might be relevant in the context of other projects. For example, IDEF1X, which concerns relational database design, might be added to the CommonKADS/IDEF3/IDEF5 framework in applications with large database storage requirements.

Creation of Explicit Models

Overview of Explicit Models Design

Explicit Models Design (EMD) (Edwards, 1990; Edwards, 1994; Edwards and Hendy, 2000) is an evolving development approach that seeks to make explicit the knowledge required by intelligent software systems. The approach compartmentalises software knowledge into five distinct, interacting models:

- **Task Model**, containing knowledge (beliefs) about tasks being performed;
- **User Model**, comprised of knowledge (beliefs) relating to the user's abilities, needs and preferences;
- **System Model**, consisting of the system's knowledge (beliefs) about itself and its abilities;
- **Dialogue Model**, containing knowledge (beliefs) related to communication among human and software agents;
- **World Model**, representing knowledge (beliefs) of the world relevant to the purpose of the software.

The resulting modular organisation offers a rich set of possible approaches for a system to assist a user. Challenges for EMD lie in how to decompose knowledge into the various models and how to co-ordinate the knowledge among the models to build effective support systems.

Plan recognition and plan generation are two additional processes that operate within the EMD framework to enhance the software's ability to support the user. Plan recognition seeks to establish the current goals of the user in the context of a larger plan. Plan generation is used by the system to develop strategies to accomplish its goals. Those techniques, and the individual Explicit Models, are described below in the context of the UAV/UCAV generic framework.

EMD's Contributions to the Generic Framework

While CommonKADS provides techniques for knowledge modelling, it does not offer methods for user modelling and that is the contribution EMD makes to the generic framework. CommonKADS could accommodate other user modelling approaches, but EMD offers an explicit, compartmentalised representation of knowledge that is not present in those approaches. Thus, EMD adds to CommonKADS a complete picture of the user, the system and the virtual world in which they interact, which is of considerable importance in a complex system such as that required for UAV/UCAV control.

Within the generic framework, EMD offers a means of subdividing the content of the CommonKADS Knowledge Model into components. Specification of all models must be done in consultation with subject-matter experts.

Task Model

The Task Model contains knowledge relating to the tasks being performed by the user, represented as a hierarchy of actions, goals and plans. At the lowest levels of the hierarchy are primitive interface **actions**, such as button clicks and menu selections. EMD recognises that each deliberate interface *action* carried out by a user is in support of a particular **goal** and that actions may be expressed in the terminology of such *goals*. For example, if a user clicks on an "OK" button, the system can infer that the user's *goal* was, "to click the 'OK' button." While the system can easily infer that low-level *goal* from the simple act of clicking an OK button, it is typically much more difficult to establish a higher-level purpose unless additional *actions* are observed. That process is described in the "Plan Recognition" section below.

Above the primitive actions and their associated low-level goals in the hierarchy are **higher-level goals**, which can be achieved only by satisfying one or more primitive goals. Higher-level goals are typically associated with what are commonly known as **tasks**. For example, in the UAV/UCAV control system, such goals could include, "to reduce altitude to 10,000 feet," or, at a higher level, the goal (task) would be, "to complete the mission."

A path from a terminal node of the tree up to a higher-level goal constitutes a **plan** for accomplishing that goal, and there can be many possible plans for satisfying a given high-level goal. Plan recognition enables the system to determine which of those plans a user is pursuing (see "Plan Recognition" below) and plan generation (see the "Plan Generation" section) permits the system to select a course of action from its available plans, or to recommend a series of actions for the user to satisfy an inferred high-level goal.

The tracking of user interface actions and inference of associated goals provides the system with a basis for understanding what a user is trying to accomplish and for helping that user in ways that are both relevant and useful. The system's ability to deduce user goals would be an essential part of any knowledge system for UAV/UCAV control and EMD provides effective methods for designing such a system.

For the user, plan generation has to do not only with the performance of a task but also with the context in which that task takes place. In part, that means knowledge of system support capabilities, which help determine a course of action. Plan recognition deals with understanding the system's intentions in offering support or with those task-related actions the user allocated to the system at some earlier time.

The EMD Task Model is constructed by extending the CK Task Model using PCT-based hierarchical goal analysis (see the section, "Hierarchical Goal Analysis of Tasks in the Agent Network"). While the CK Task Model serves as an effective starting point for a high-level organisational analysis, it does not offer a sufficient degree of low-level detail for implementation of an intelligent, adaptive interface. The EMD Task Model adds that level of detail to the modelling process.

System Model

The System Model is composed of the system's knowledge about itself, its abilities and the means by which it can assist users. Like the Task Model, the System Model also contains a goal hierarchy, describing the tasks, goals and plans that the system can carry out in support of the user. Those goals are characterised as **system support goals**. Within the generic framework, systems will be composed of multiple agents, thereby requiring a more complex decomposition and distribution of knowledge. Where multiple system agents are involved, the System Model will be partitioned to store a goal hierarchy for each agent.

In the generic framework, the System Model task hierarchy includes high-level goals, such as, "to assist the user," which would be decomposed into sub-goals, such as those associated with assuming control of sub-systems it had been assigned, monitoring mission status and helping the user to complete his or her tasks.

The level of assistance provided by the system is determined by the automation level, specified as part of a **contract** between the user and system. Users may request any of six levels of assistance ranging from no automation to a fully automated and autonomous help system. For more information, see "The PACT Approach and Automation Levels" section.

User Model

The User Model is comprised of knowledge about the user's abilities, needs and preferences. That information is obtained in three ways:

- from information volunteered by the user;
- from the results of system requests of the user;
- from system monitoring of user's activities.

It is worth noting that the system should be able to identify a particular user so that it can maintain a unique profile for each user. Unless that can be done, the system is reduced to providing information that is often too general, repetitive or useless.

Information volunteered by a user often occurs in the context of specifying options and preferences to the system. It is important that users be able to specify preferences in order to facilitate efficient use of the software, and that is especially useful in domains such as UAV/UCAV control in which operators can experience high cognitive workloads.

There are several ways in which systems can construct user models by explicitly asking questions of a user. For example, if the system has determined which task a user is pursuing, it could enquire whether assistance is needed in carrying out that task. The system also might ask if the user is aware of more efficient plans for accomplishing the task. Finally, if the system cannot determine a user's current plan, it may seek clarification on the user's intentions.

In a truly intelligent system, User Model knowledge is acquired indirectly by monitoring user activities in a Task Model. If the system observes the user carrying out a particular plan, it is assumed with a fairly high degree of confidence that the user understands that process. The system's confidence increases as the user is observed to repeat the procedure.

If the system determines there is a high probability that a user lacks certain required knowledge, that signals a need to offer the information. The system must be able to gauge the importance of communicating the information in order to establish a method for doing so. For example, if human safety is at risk, the operator might need to be informed immediately. In contrast, advice on carrying out tasks efficiently might not be presented until the operator has completed the current task or the entire mission.

World Model

The World Model contains the software's knowledge about the external world: the objects that exist in the world, their properties and the rules that govern them. Those rules can take on a wide variety of forms, such as physical (e.g., the principles governing the effect of aircraft controls on flight parameters), psychological (e.g., rules describing human behaviour in situations of high cognitive workload) and cultural (e.g., rules associated with tactics commonly employed by adversaries).

In the case of the UAV/UCAV control system, the World Model likely would contain:

- geographical knowledge of terrain and political boundaries or access to such knowledge maintained by a Geographical Information System (GIS);
- principles of flight dynamics and aircraft control and their implications;
- information about the enemy, including targets, equipment and tactics.

In the "Ecological Interface Design" section below, needs are described for mission scenarios and rules of engagement to be stored in a knowledge base. Such information would be suitable for the World Model.

Dialogue Model

The Dialogue Model contains knowledge about the manner in which communication takes place among user and system agents. Such communication would involve interaction between the user and system and among other system agents.

Because there will be many system agents in the UAV/UCAV system, it will be essential to specify a common language and protocol for them to communicate. In addition to that, effective user-system and system-system collaboration will require the explicit representations of communication provided by the Dialogue Model.

Finally, the Dialogue Model must be designed to permit feedback in agent communication since EMD and PCT rely on the presence of effective feedback. (See the discussion of PCT (Perceptual Control Theory) in the next major section.)

Plan Recognition

The ability to recognise user plans is an important element in EMD and enhances the system's "awareness" of what a user is trying to accomplish so that it can decide how best to offer assistance. It is the infrastructure of intelligent, adaptive aiding. One implementation of an EMD system has incorporated techniques from the COLLAGEN approach developed by Lesh, Rich and Sidner (1999).

COLLAGEN uses a "recipe" approach whereby plans that the user may be pursuing are assembled from **plan fragments**. When an interface action is observed by the system, the *fragments* are assembled to form plans that explain why the user performed that action, and there may be many possible plans. As further user actions are observed, the choice of plans that encompass that series of actions diminishes, leading to a more accurate determination of the user's true plan.

The EMD implementation of COLLAGEN mentioned earlier does not provide all the functionality of that technique. For example, COLLAGEN provides for clarification dialogues in which the system

prompts the user when there is ambiguity in determining the current plan. That ability could be useful in the UAV/UCAV system, but the system would need to ensure that such prompts did not disrupt the controller's operation of the aircraft. Such communication should be represented in EMD's Dialogue Model.

Plan Generation

Plan generation is the process by which the system develops strategies for accomplishing its goals to assist the user. It is based on System Model knowledge of a hierarchy of available support goals and plans, Task Model knowledge of the user's current goals and plans and User Model knowledge of the operator's preferences and abilities.

Plan generation in the generic framework would seek to construct the most effective plans for the system to help the controller, e.g., by taking over control of sub-systems under circumstances for which that control had been assigned or monitoring mission status, depending on the selected level of automation.

The processes of plan recognition and plan generation also can be associated with activities in Perceptual Control Theory (PCT) (see the next major section for a discussion of PCT), whereby plan recognition is associated with the perceptual input to hierarchies of system control loops and plan generation forms the behavioural output of similar hierarchies. Those parallels bridge the EMD and PCT techniques in the generic framework.

Feedback

The concept of **feedback** is important in EMD for establishing mutual understanding and support between the user and the system, enabling one agent to inform another of its goals, plans and knowledge. Feedback can assume multiple forms, both explicit and subtle.

Explicit feedback can occur in the form of dialogues among agents. For example, the system may ask a user whether he or she is familiar with a particular concept. The user's response constitutes feedback to the system, providing knowledge for the User Model and therefore enabling the system to offer more appropriate assistance. Similarly, a user might ask the system to explain its last action, particularly if that action was performed on the system's own initiative. The response from the system is feedback that gives the user a better understanding of how the software operates. The communication of explicit feedback among agents is governed by the Dialogue Model, which must be designed to support exchanges among agents involving the provision of feedback.

A less overt form of feedback arises in the form of system support goals and user goals. For example, if a user's goal is to open a window in the software interface, the system will have a corresponding support goal to display that window. The display of that window constitutes feedback to the user that the goal of opening the window in the virtual environment has been achieved. Representations of user goals also are important forms of feedback since they are the primary means by which the system knows and learns about a user. Detecting user goals is detecting feedback in that those goals implicitly inform the system of a user's plans, abilities and preferences.

The role of feedback in Perceptual Control Theory (PCT) control loops is discussed in the next section. Like plan generation and recognition, feedback helps support the integration of PCT and EMD components in the generic framework.

The Role of Perceptual Control Theory

Overview of Perceptual Control Theory

Another theoretical approach in the generic framework is Perceptual Control Theory (PCT). While the IDEF and CommonKADS methodologies provide frameworks for approaching the design and implementation of the system, PCT and Explicit Models Design (EMD) have the potential to influence how the system functions within that implementation framework: how it determines the goals a user is trying to achieve, the plans for achieving those goals and how it can assist the operator most effectively. The techniques complement one another and together provide an opportunity to form a comprehensive approach that combines the strengths of the individual components.

PCT is founded on notions from control theory, in which closed-loop, negative-gain, feedback systems can be used to build powerful models of goal-directed behaviour and for implementing complex systems. Figure 1 shows the basic PCT model structure (from Hendy, Beevis, Lichacz and Edwards, 2002). The fundamental idea of the theory is that humans are closed-loop systems that work to achieve their goals by exhibiting behaviour that acts to control their perceptions. Raw sensory input (sensation) from the world (**W**) is interpreted as human perception (**p**). That perception is then compared against a reference signal corresponding to the person's goal (**g**) and any discrepancy is reflected in an error signal (**e**). (See discussion on the origin of reference signals in PCT below in, "Hierarchical Goal Analysis of Tasks in the Agent Network.") If an error signal is present (i.e., the goal is inconsistent with the current perception of the world), the result is a change in behaviour intended to minimise that error. The effect on the world is interpreted again as perception and the looping continues.

There are two additional items presented on the next page in Figure 1 that have not been factored into the description above. The inner loop mirrors the outer loop described above, except that it reflects mental rehearsal (simulation) of responses to perceptual error. Thus, the human imagines how to deal with a variety of perceptions without actually modifying behaviour that would be apparent to an outside observer. When the PCT model is applied to the case of a goal-directed (machine) agent, the inner loop corresponds to the reasoning capabilities of that agent.

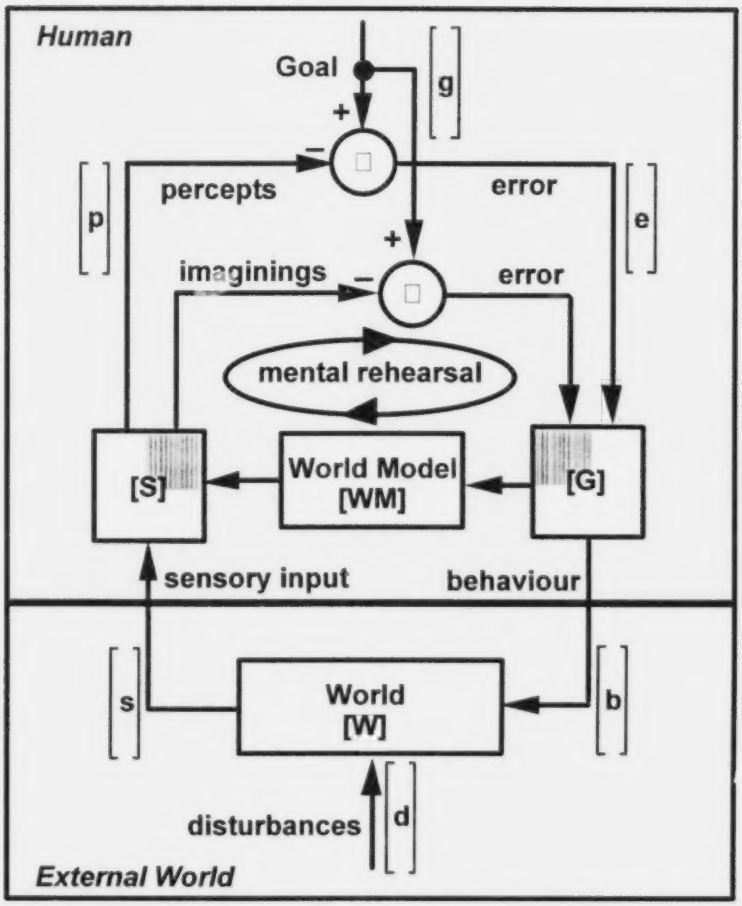


Figure 1 The Perceptual Control Theory Model

The remaining item in Figure 1 to be discussed is the notion of disturbances (**d**), which are external factors that influence the state of the world. One of the strengths of PCT over competing theories is that it explains how humans can control systems that are subject to a wide variety of external influences. Thus, a person's behaviour adapts to accommodate the disturbances in order to achieve consistent results. Because of the variable nature of disturbances, the method of accomplishing a task can be different each time it is performed, but the result will always be the same.

PCT control loop concepts are frequently illustrated using the example of a person driving a car (Powers, 1990a). Here, the driver works to control the perception of the car's position on the road so that the car remains centred in its lane. If the driver's perceptions indicate that the car is too far to the left, the resulting error signal causes the driver to compensate by turning the steering wheel to the right. The further the car is from being centred in the lane, the greater the error signal, and the more the wheel must be turned to compensate. That property gives control systems their error-correcting

characteristics and enables them to achieve a desired goal state very quickly. Disturbances in the car example could take the form of crosswinds, flat tires or ice on the road.

It will be shown that PCT has much to offer the generic framework. The primary benefit of such an approach is that it offers the rigours of a formal theoretical framework within which to design the system. The capabilities that PCT can provide to the system could be achieved using *ad hoc* techniques, but PCT offers an additional theoretical underpinning that those other techniques do not.

PCT is one of many theories of human behaviour. Elements of cognitive theories, of which PCT is one, already are present in Explicit Models Design. They include techniques for decomposing user and system goals and recognising user plans based on observed actions. In addition to providing an explanation and model for goal-directed human behaviour (see discussion on the origin of reference signals in PCT in, "Hierarchical Goal Analysis of Tasks in the Agent Network") and being equally applicable to describing goal-directed human and machine behaviour, a key advantage to integrating PCT concepts with the UAV/UCAV control system is its suitability for programmatic implementation given its engineering origins in control theory.

The ways in which PCT can contribute to the UAV/UCAV control system fall into two basic categories:

- performing hierarchical goal analyses;
- using PCT principles in the algorithms of the system.

Hierarchical Goal Analysis of Tasks For UAV/UCAV Control

Hendy et al. (2002) offer a new method of Hierarchical Goal Analysis (HGA) using principles from PCT, and that technique has the potential to produce a robust and complete task and goal decomposition for UAV/UCAV controllers. Tasks, goals and plans are important terms in this and other related approaches. As described earlier, in the section on Explicit Models Design, tasks typically correspond to high-level goals, such as, "to reduce altitude to 10,000 feet," or, "to complete the mission." More specifically, a task can be characterised as a high-level goal and a set of plans for achieving that goal, where a plan corresponds to a series of low-level goals required to achieve a particular high-level goal.

The new method of HGA presented by Hendy et al. examines the goal of an agent as a desire to achieve a certain perception. One aspect of characterising goals in that way is the necessity to consider only those goals that exert influence over some variable, which could be either an external property of the world or an internal mental state. Thus, each goal in the hierarchy has the capacity to be assigned to some agent in the system, either human or machine, and modelled using a PCT control loop. It does not make sense to include goals that do not have that property since they are unlikely to be satisfied (see the discussion of obstacles to goal achievement at the end of the "Using PCT Concepts in UAV/UCAV Control System Implementation" section below). So, examining goals from the viewpoint of manipulable variables tends to maximise the enumeration of assignable goals and minimise attention to those goals which cannot be satisfied.

Hendy et al. (2002) cite an example of the application of the technique to a goal analysis of the Canadian Land Forces Tactical Battlefield Command System. A previous analysis using the established technique of Mission, Function, Task Analysis (MFTA) (IEEE-1220, 1994) failed to identify numerous goals that were present in the system, and which were subsequently identified using the PCT-based approach. That study also showed that the theoretical approach of PCT allows for the consolidation of separate MFTA functions and tasks in HGA, thereby illustrating the power of

PCT to simplify the process while providing confidence in the completeness of a PCT analysis. A similar analysis can be done by applying HGA to the UAV/UCAV system.

Hendy et al. (2002) have shown that the PCT-based HGA technique permits two additional analyses to be performed. The first is a stability analysis which identifies possible conflicts among multiple human and machine agents acting on the same system. The second is an analysis of information flow up the hierarchy, which can influence feedback in the system, affecting error-correction at higher levels. Traditional HGA systems analysis techniques consider only the downward flow of information in the hierarchy. In the case of the UAV/UCAV system, it is critical that information in the form of system and user goals and sub-goals be able to flow freely in both directions. Stability and information flow analyses could contribute to the generation of a robust goal hierarchy for the UAV/UCAV control system.

Reorganisation and Machine-Learning

PCT does not emphasise the origins of reference signals (or goals) in control loops, but the theory does not ignore that matter. Mechanisms have been proposed whereby innate behaviour in humans forms early reference signals, which are then modified via a process of reorganisation, resulting in learned behaviour (Powers, 1990b). A machine implementation of a PCT control loop must begin with a "hard-wired" reference signal in order to be functional, but applying machine-learning techniques could enable the extension of hard-wired goals to produce a more robust and adaptive system.

Attention would also need to be given to circumstances under which machine-learning could take place. PCT postulates that learning in a human occurs only when existing control mechanisms are inadequate to eliminate error signals. When such circumstances arise, the process of reorganisation begins, introducing random behaviours into the control loops. That element of randomness gives rise to behaviours that would not otherwise be present and creates opportunities for chance occurrences to eliminate the error signal that initiated the reorganisation. When the error signal has been reduced to zero, the reorganisation stops, leaving in place control loops that are better able to deal with similar error signals in future.

For example, an infant does not have an innate ability to reach for an object, and so may perform random arm movements until contact is made with the object. The PCT view is that the random motions signify that reorganisation is occurring and cessation of that motion upon contacting the object indicates that the learning process has stabilised, leaving in place control loops that facilitate future attempts to grasp objects.

If reorganisation were occurring continuously in a system, the random behaviours would interfere with the normal operation of the control loops, so, to avoid that, reorganisation only takes place when existing mechanisms are not adequate to eliminate error signals. If the concept of reorganisation were applied to enable machine-learning in a PCT system, a similar approach would be required to limit the introduction of random behaviours to situations where error signals could not be eliminated by any other means.

It should be noted that many details of how learning takes place are missing from the concept of reorganisation in PCT. The theory does not address the specifics of how randomness is introduced: what behaviours are altered and to what extent. That has bearing on the practicality of implementing machine-learning using PCT concepts, depending on the level of abstraction involved. For example, a low-level activity, such as regulating the speed of a car with a cruise control, likely could be "learned" by a mechanical system by producing random adjustments to the accelerator pedal in response to perceived changes in speed. Such a system probably would learn very quickly the

amount of gas required for an appropriate adjustment in speed. In contrast, higher-level goals, such as those that would be present in the agent network, have many more degrees of freedom in their range of possible behaviours, making it much less likely that a system could randomly "learn" how to achieve a high-level goal.

That inability to make a machine "learn" to achieve high-level goals through traditional PCT reorganisation rules out the possibility of using purely random processes to enable learning within the generic framework. However, other methods of learning can be integrated with the PCT approach to increase the effectiveness of reorganisation in the control loops. For example, it is theorised in PCT that perceptions in human control loops can be stored as episodic memories and that such memories are associated with temporal cues. There is a strong parallel with the logging of perceptions within Explicit Models where, for example, the perceived actions of a user are stored with associated time stamps.

Those stored perceptions provide a basis for a reorganising system in the generic framework to observe the behaviours that have given rise to certain perceptions and thereby "learn" how best to behave in the future. For example, the system might observe that certain adaptive aiding techniques are more effective than others at training a particular user. Similarly, that user might tend to ignore another aiding method. Thus, having "remembered" that, the reorganising mechanism could favour the generation of system behaviour that is most effective for that user.

Using PCT Concepts in UAV/UCAV Control System Implementation

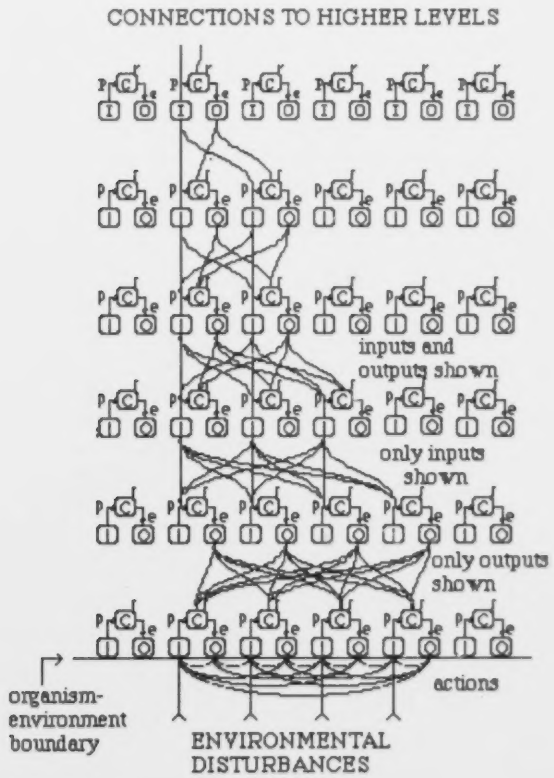
Once a Hierarchical Goal Analysis has been performed on tasks carried out by the agent network, the next step is to consider how best to incorporate PCT concepts into the operation of the system, and some of those notions were introduced in the preceding section. The idea of creating a goal-driven control loop within the agent network is consistent with the monitoring of user actions that are addressed by Explicit Models Design (EMD). Also, perceptual and behavioural elements can be seen to have parallels with plan recognition and plan generation components, respectively.

The action monitoring mechanism in EMD can be adapted to take the form of a goal-driven control loop modelled on PCT principles. Other software systems have been developed that make use of PCT control loops and, typically, such systems involve the control of low-level activities, such as the operation of motors based on sensor readings in a robot. Control theory works very well in those applications because it can allow the system to accomplish its goals without requiring extensive modelling of the outside world. For example, a PCT-based walking robot (e.g., Kennaway, 2000) need not be aware of the contours of the terrain it is negotiating, it need only be able to perceive the angles of its leg joints and whether or not each foot is in contact with the ground. That limited awareness enables it to move over a wide variety of terrains and obstacles. The continuous nature of measurements such as leg angle is also well suited to the control system approach because it allows full use of the intrinsic error-correcting properties of such systems. Those properties of PCT can lead to very simple, versatile and powerful systems.

To date, PCT has not been used for implementing systems with a "higher level of awareness," such as the proposed agent network. Unlike the robot example above, the agent network does not involve continuous measurements and instead deals with discrete notions, such as whether or not the user requires assistance. Application of perceptual control loops to the UAV/UCAV control system provide an ideal opportunity to illustrate the suitability of PCT to higher-level applications with discretely valued attributes while achieving at least some of the aforementioned benefits of the theory.

Control Loop Hierarchies

Powers (1989) describes how PCT systems can be implemented as a hierarchy of control loops, wherein the output of the higher levels determines the reference signals at the levels below and the perceptions at lower levels feed the inputs at the levels above. Figure 2 illustrates how the inputs and outputs of control loops are linked in the hierarchy. Due to the large number of connections, some have been omitted from the diagram for clarity.



Source: Adapted from Powers (1989a, p. 278).

Figure 2

Figure 2: Hierarchy of control loops.

Powers (1990a) also proposes a model of human behaviour as an eleven level hierarchy, where the lowest levels deal with the most basic human perceptions of sensory stimuli and the highest levels address abstract reasoning and creative processes. Machine systems that use PCT, such as the robot

mentioned above, typically are not implemented to mirror the eleven human levels of perception, but instead follow a decomposition that is tailored to the specific needs of the machine and the resulting hierarchy usually operates only at low levels. In the case of the Kennaway robot, only two rudimentary levels control the mechanics of limb movement.

In the case of the agent network, the most relevant perceptions (beliefs) will occur at a level comparable to the ninth level in Powers' hierarchy, known as the **programme** level. At that level, systems are able to infer logical relationships among perceptions, such as determining how a series of user actions fits into a broader plan and the implications of the pursuit of that plan. The *programme* level encapsulates perceptions from lower levels, such as the observation of sequences and relationships among input values.

To implement control loops in the UAV/UCAV control system, the loops need to be assigned a hierarchy of goals, modelled largely at that level and described in the "Hierarchical Goal Analysis of Tasks in the Agent Network" section above. Those control loops have as input the actions carried out by the user. The actions serve as a basis for system perceptions about the user's need for assistance and that constitutes feedback in the loops. Later, it will be shown that a complementary process exists: system actions form the basis of user control loops.

Agent Network Goals and Sub-goals

If a high-level goal of the system is, "to have the perception (to believe) that the user is performing the current task in a way that requires no intervention by the system," a control loop would need to be monitoring activities at the interface (perceptual input) to detect the satisfaction of that goal. Support for such a control loop means that the system must infer and represent a belief that the user is engaged in a particular task based on perceived user activity at the interface. In order to infer such a belief, the system must have knowledge of the structure of goals and sub-goals necessary for operators to perform the task.

The high-level system goal can be further decomposed into sub-goals concerning types of user activity that could suggest to the system that some form of assistance may be necessary. System control loops should identify a potential need to offer help when it is perceived that the user is not performing a task:

(i) correctly;

(ii) completely;

(iii) consistently;

(iv) efficiently;

(v) safely.

That gives rise to five sub-goals, each with its own hierarchy of sub-goals. That same five-part taxonomy is being developed in the context of another project of a generalised, intelligent system to provide help to software users. In that application, the system's decision to offer help is based in part on an assessment of user needs according to whether tasks are being carried out in compliance with the five criteria above.

One of the strengths of PCT is rapid error-correction as a result of making behaviour changes commensurate with the degree of error between the perceived and a goal state. To return to the

illustration of driving a car, the amount by which the car deviates from the centre of the lane determines the magnitude of correction applied to the steering wheel, thus enabling a faster return to the goal state. In the case of the UAV/UCAV system, it may be desirable to examine the **degree to which the user is in need of assistance**, such that some perceived user actions could trigger a more immediate response, while others might suggest that the system should wait for further input. **Feedback from the user to the system (represented using control loops)** would be crucial to enabling the system to determine what level of assistance was warranted.

In PCT terms, plan recognition can be seen as a perceptual activity with sensory input from the outside world processed to form the perceptions that are compared with the goal signal. That allows the raw data from a person's sight to be interpreted as, for example, a measure of how far a car is off-centre in its lane. In the UAV/UCAV system, raw data about the user's actions can be processed using plan recognition to identify the user's broader task, which then is interpreted in the context of the system's adaptive aiding control loops.

The generic framework specifies that system-generated plans to assist users will be incorporated into the behavioural components of control loops. Plan generation mechanisms will examine perceptual error signals and formulate appropriate behavioural responses to correct them. The plan generator should take into account the magnitude of the error signal in determining the optimal behaviour for providing assistance under the circumstances, whether it be automatic or involve querying the operator on how to proceed.

PCT Advantages in the UAV/UCAV Control System

PCT has other advantages that have bearing on the control system. PCT is scalable, so that the degree of hierarchical decomposition can be tailored to a specific application of the technique. For example, a system controller's goal hierarchy could be decomposed into very low-level activities, such as firing neurons to trigger muscle movements to operate the mouse. Although of little or no concern in many applications, attention to such details could tie into physiologic monitoring devices connected to UAV/UCAV operators (see the "UAV/UCAV Agent Hierarchy and PCT" section below).

PCT theory works well in a collaborative setting, where multiple agents are acting on a system, because of its inherent error-correction mechanism. The error-correction arises because control loops continuously receive feedback about the state of the environment and adjust their behaviour to compensate. While that mechanism deals effectively with external disturbances, it can also automatically compensate for system changes due to the actions of another agent, which would favourably impact performance in a multi-agent system.

Along with implementing the UAV/UCAV interface using control system techniques, the user could also be modelled in terms of PCT concepts. For example, a property of goal-driven control systems is that they can repeatedly exhibit the same behaviour in a dysfunctional attempt to correct a perception that is inconsistent with the goal state. That arises as a result of the looping nature of the system and the fact that the behaviour is not altering the perception. It may also be associated with reorganisation processes associated with problem-solving on the part of the user. In the case of a PCT-modelled UAV/UCAV operator, the observation of a repeated set of actions could inform the system that he might not be accomplishing his goal and is therefore in need of assistance.

Modelling the user as a PCT control loop requires that system activity serve as input to the feedback loop. Forming the basis of that feedback would be the "system support actions," which are carried out by the system in support of the goals of the user. In that way, a symmetric relationship is established, whereby system actions form the basis for the user feedback loop, while user actions drive the system feedback loop.

A further possible application of PCT concepts could include the creation of an explanation mechanism so that users could obtain feedback from the system on its understanding of current activities. That could occur in a level in the hierarchy above the main system goals, where a higher degree of perception would permit generalisations on the programmes involved. The intention of such an explanatory mechanism would be to further mutual understanding about significant aspects of the mission between the user and system. Linguistic elements would need to be incorporated to translate perceptions and behaviours into a form intelligible by the user. Those need not begin to approach full natural language competence. Finally, a very important aspect of the meta-analysis of programmes is that such an analysis could form the basis for the alteration of the programmes themselves, i.e., for incorporating learning methods into the system that would change the system. In other words, they could support the construction of programmes whose job it was to alter other, existing programmes, which is at the heart of the learning process.

Disturbances and Replanning

Disturbances in perceptual control loops are often illustrated by low-level activities such as steering a car to maintain its position in the centre of a lane. The disturbance of a crosswind or a tilted road requires a simple adjustment to steering behaviour to eliminate the error signal, but other disturbances could have ramifications for higher level goals. For example, a fallen tree blocking the road constitutes an obstacle to achieving perceptions consistent with the reference signal, since no amount of steering will allow the car to get beyond the tree. Such a situation could necessitate stopping the car, turning it around and finding an alternate route or by invoking known alternative routes and selecting one. The perception of a need for **replanning** must occur at a higher level of abstraction than the steering control loop provides, because a low-level steering control loop would be incapable of eliminating the error signal. Therefore, higher level loops must address replanning but PCT does not offer much guidance as to what that replanning would entail. While it is second-nature to humans to adapt to the emergence of an obstacle, it is less trivial to incorporate replanning techniques in a PCT-based machine. Since the UAV/UCAV system will require an ability to replan based on changing conditions, work on the intelligent interface should explore the possible inclusion of a high-level control loop for perceiving a need for replanning and for generating behaviour that would accomplish that replanning.

Software Agent Paradigm

Overview of Agent-Oriented Development

An autonomous software agent is a programme with the ability to sense its environment and to act on that environment over time to achieve some purpose and to influence what it will sense in the future (Franklin & Graesser, 1996). Agents can be distinguished further based on their characteristics, for example, **communicative** agents can interact with other agents or people; **adaptive** (or **learning**) agents can alter their behaviour based on past experience; and, **mobile** agents can move themselves to other machines.

The agent-oriented development paradigm offers several advantages that were not addressed by earlier object-oriented approaches, including:

- increased modularity;
- enhanced reusability;
- improved organisational effectiveness;
- increased speed;
- increased reliability;
- better distribution.

Programme modularity is a persistent goal in software development to improve the organisation and reusability of software components. While traditional object-oriented design allows programmes to be sub-divided into modules, the agent paradigm takes that a step further by separating software functions into an array of individually-acting programmes. Those components can be reused and combined with other agents to create new, and potentially very different, pieces of software. Modularity can also serve to improve the organisational effectiveness of software, whereby a suitable division of labour among software components can lead to improved functionality, e.g., by enabling the software to find better solutions or to avoid errors in the process.

Agents also can improve the speed at which software operates since they inherently operate independently. Their concurrent operation allows each agent in the system to act as soon as it perceives a reason to do so, and without waiting for other components to complete their tasks. While some situations require one agent to finish before another can begin, there are also circumstances in which the first agent can provide partial results to the second agent prior to completion of the task, thereby increasing efficiency. Further speed gains can be made through distribution of agents on multiple platforms (see below).

The agent approach can increase reliability through redundancy. When a single programme is running on a network, it may not be able to operate at all times as a result of network instability or system maintenance. In contrast, when multiple instances of an agent are operating in different locations on the network, they can all be working toward a common goal without the same susceptibility to problems on the network. In such situations it is important that the individual agents be able to communicate with one another to co-ordinate their activities. Co-ordination also requires

that agents maintain an understanding of what other collaborating agents are doing, as well as strategies for determining the best actions to achieve the common goal.

Distribution is another inherent advantage to agent-based software. The distribution of agents over a network can allow software to make better user of available processing power by dividing a large task into smaller sub-tasks handled on separate machines. That use of agents is especially important for computationally intense tasks, such as those carried out within a knowledge-based system. As with the previous example, distributed agents require the ability to co-ordinate activities through network communication, awareness of other agents' activities and strategies for accomplishing the common goal.

The ideas for software agents have been around since the late 1970s, but the approach did not enter into widespread use until the 1990s. Much of the initial work on agents came out of the (distributed) artificial intelligence (AI) community because autonomous agents naturally lend themselves to AI approaches. More recently in AI, Wooldridge and Jennings (1995a, 1995b) had a significant role in fostering substantial interest in autonomous agents as a software development paradigm. With the current widespread adoption of Internet and intranet technologies, the opportunities for using agents continue to expand.

The success of the agent paradigm has led to its use in a wide range of fields, including AI, knowledge management, human-computer interaction and Internet-based application development. The last category includes information search and retrieval as well as personal software agents responsible for scheduling or making purchases on behalf of a human user.

Agents offer numerous advantages for the generic framework. Intelligent, autonomous agents are ideally suited to taking over some tasks for human operators, serving to support the goals of reduced manning and enhanced performance in a complex environment. Also, the redundancy and distribution across systems leads to improved reliability and safety because, in the event of communication breakdown, mechanisms are in place to compensate.

Agents in CommonKADS

The CommonKADS (CK) methodology (discussed in the "CommonKADS Methodology" section at the beginning of this report) is entirely consistent with the use of software agents. The Agent Model in the methodology allows for systems with multiple human and software components.

The developers of CommonKADS have provided an example to illustrate the feasibility of designing a distributed multi-agent system using the methodology (Schreiber et al., 2000). The example begins by noting that as a consequence of the European deregulation of the electric utility industry, the business of "generating, distributing and billing customers for kilowatt hours…is being transformed into offering different kinds of new value-added customer services" (pp. 220-221). In Sweden, one of the anticipated service applications is that the electric nodes will act as intelligent agents. CK developers refer to such a system as, "Homebots." In that system, each electricity customer would have a software agent negotiating on his or her behalf with software agents representing the power company. The system would operate continuously, allowing both consumers and suppliers to obtain favourable rates on electricity according to current supply and demand constraints.

While that example involves a more distributed network of agents than would be suitable for the UAV/UCAV control system, it demonstrates the suitability of CommonKADS for engineering and managing systems with multiple agents.

A Multi-Agent System extension of the CommonKADS methodology (MAS-CommonKADS) has been proposed by Iglesias, Garijo, González, and Velasco (1996). The methodology was developed to add specific agent-related constructs, including those relating to: (a) inter-agent communication; (b) the division of tasks among individual agents; and, (c) the implications for implementation of multi-agent systems.

The Communication Model in CommonKADS is primarily focussed on interaction between the user and individual system agents, with little attention paid to communication among the system agents themselves. To address that issue, MAS-CommonKADS incorporates a Co-ordination Model, which specifies how messages are exchanged, the communication protocols and the abilities that each agent has for interacting with the others. Because of the many commonalities between the Communication and Co-ordination Models, the latter could be treated as an entity within the former.

The division of tasks among agents is an important consideration in the MAS-CommonKADS approach. The physical locations of agents and connections among them can influence the assignment of responsibilities to each component. The division of tasks also will affect the knowledge requirements of each agent.

The multi-agent approach also influences the construction of Design Model specifications. Consideration must be given to network facilities, transfer protocols, encryption and authentication according to hardware and software constraints. MAS-CommonKADS addresses those issues.

Agents and the IDEF Standards

The IDEF Standards do not address intelligent software agents explicitly, but IDEF techniques are well suited to application in the context of an agent-based system. The "IDEF3: Process Modelling" standard can be used for describing the activities carried out by each agent in a system, particularly where complex temporal constraints must be represented and the UML approach used by CommonKADS is not adequate. "IDEF5: Ontology Modelling" is suitable for developing ontologies to represent the knowledge requirements of individual intelligent agents. See the earlier section on "IDEF's Contribution to the Framework" for more detail on IDEF techniques.

Agents in Explicit Models Design

Explicit Models Design (EMD), described above, also supports multi-agent system development. EMD recognises the roles of the User and System as agents and allows for the involvement of both multiple human users and system agents with each represented by its own User or System Agent Model. The ability to have multiple operators is useful in the context of UAV/UCAV control. For example, there may be advantages to having two operators together controlling, say, ten UAVs, as opposed to each operator independently controlling five aircraft. The manning requirements are the same, but there is the potential for increased flexibility when one operator must devote full attention to one or two UAVs and the other operator can attend to the rest. In those situations, system agents will be essential for assuming some responsibilities to help ensure that the operator overseeing a larger than usual number of aircraft maintains a manageable workload.

The EMD Dialogue Model provides a framework for describing communication among multiple human and system software agents. That model allows for various modes of communication, including the following, relevant to UAV/UCAV control:

- the user instructing the system to take over a task;

- the system prompting the operator for clarification feedback about user goals;
- communication among system agents to co-ordinate activities.

As indicated earlier, the System Model in EMD represents the system as a set of co-operating autonomous agents. Provisions are also made for external agents to play a role supporting the goals of both human users and system agents. That technique has been demonstrated within the context of the *LOCATE* workspace layout tool where agents monitor certain user actions that serve as triggers for information-retrieval activity from selected sites on the Internet (Edwards, Scott and Hendy, 2003).

Hierarchical Goal Analysis (HGA) of Tasks in the Agent Network

In the same way that hierarchical goal analysis (see the preceding major section for a discussion of HGA) can be applied to the task goal hierarchy for the UAV/UCAV controllers, it can also be used in an analysis and design of a goal hierarchy for the network of intelligent agents that will assist human users. That would offer the same benefits described earlier: a thorough decomposition and the ability to conduct stability and information flow analyses.

The highest-level agent network goal could take the general form: "to perceive (believe[2]) that the user is able to perform the current task unaided," i.e., that the user does not require assistance from an intelligent agent. That could be decomposed into sub-goals regarding the perception of different signs that a user needs assistance, e.g., "to perceive (believe) that the user is performing the current task in the most efficient manner." An error signal would result when inefficiencies in the user's actions were detected that could require an agent to intervene.

UAV/UCAV Agent Hierarchy and PCT

Hou (2003) has proposed a hierarchy of agents monitoring user needs. Those agents would monitor cognitive factors, such as workload and situation awareness, as well as physiologic signs, such as eye-gaze tracking and heart rate. Such an arrangement of agents lends itself to implementation as a hierarchy of control loops. The physiologic signs correspond to low-level perceptions in PCT, while cognitive factors correspond to higher-level perceptual activity.

PCT control loops are suited to implementation as agents because they share a number of characteristics. They are single-purpose entities that can perceive and act. Although each agent is autonomous, all could be linked together through control loop inputs and outputs, yielding a larger system that is far more powerful than any of its individual components.

The advantage of creating control loops as individual agents is that it gives rise to reusability and consistency within the implementation framework.

The PACT Approach and Automation Levels

Since it is unrealistic to expect that all users will require the same level of automation from an intelligent system at all times, a need exists for users to be able to specify their requirements of the system. In the UAV/UCAV control domain, circumstances of a particular mission and operator

---

[2] In PCT, the sensory input signal from the outside world forms the basis for **perceptions** in the feedback loop; other cognitive theories characterise those *perceptions* as **beliefs** about the world.

preferences will dictate what automation level is most appropriate, and those could change several times within a single mission.

One method of handling automation levels in the air domain was developed for the Cognitive Cockpit (COGPIT) project by DERA in the United Kingdom (Taylor, 2001). Known as Pilot Authorisation and Control of Tasks (PACT), the system is based on the notion of **contractual autonomy**, in which the user and system establish an agreement, or **contract**, on the system's responsibilities. *Contracts* are made using a system of six levels, numbered from 0 (no automation) to 5 (fully automatic). Table 1 shows the levels of autonomy in the PACT approach.

Levels	Operational Relationship	Computer Autonomy	Pilot Authority
5	Automatic	Full	Interrupt
4	Direct Support	Advised action unless revoked	Revoking action
3	In Support	Advice, and if authorised, action	Acceptance of advice and authorising action
2	Advisory	Advice	Acceptance of advice
1	At Call	Advice only if requested	Full pilot, assisted by computer only when requested.
0	Under Command	None	Full

Table 1: PACT levels of autonomy.

Contracts in the generic framework should offer users the ability to set the autonomy level and change it at any time during a mission, as well as provide the flexibility to customise the provision of specific forms of assistance. That customisation should allow operators to request help at specified intervals or to ask that help be provided in a specified form (see the "Forms of Adaptive Aiding" section).

Human-Computer Interaction

Effective human-computer interaction (HCI) is essential in environments with high workloads, such as the UAV/UCAV control system. In those situations it is crucial that the operator be presented with the right information at the right time and that the interface be presented in such a way as to maximise the operator's ability to execute tasks efficiently. Of all the techniques available in HCI, one that has commanded considerable recent attention is Ecological Interface Design (EID). Given the level of general interest, EID was examined for how it might contribute to the proposed generic framework.

Overview of Ecological Interface Design

Ecological Interface Design is a framework for problem domain analysis and the design of human-machine interfaces in complex work environments that was developed at the Risø National Laboratory in Denmark by Vicente and Rasmussen (1992). The name reflects the incorporation of elements from ecological psychology, particularly the emphasis on the importance of considering the interaction of humans with their environment. While most traditional interface design approaches confine their attention to human characteristics, EID also examines how humans interact with their surroundings, taking into account both physical and cognitive factors, in the context of the complex system under control. To achieve that, EID offers concrete guidelines on interface design, with the aim of producing optimal usability and safety.

EID has been applied in a variety of areas, including nuclear power plants, industrial process control and in the medical and air domains. One of the dangers when working with such critical systems is the incidence of abnormal events. Events that are unfamiliar to system operators can be dealt with if system designers have foreseen such occurrences and implemented procedures to address them. In contrast, events that are both unfamiliar to operators and unanticipated by designers can lead to critical system failures. Even when safety is not a concern, those events can contribute to diminished system performance. As it is impossible for designers to plan for every eventuality, it is important for interfaces to convey information to operators in the most effective way possible to allow them to recognise and resolve potential problems. EID was developed to achieve that goal. Complementing its ability to address safety issues, EID also is intended to produce interfaces that are intuitive and flexible for users.

EID recognises that system reliability depends not only on the engineering of that system, but also on how humans interact with it. For that reason, the content and structure of the interface must be designed to facilitate the operator's understanding of the system being controlled. Traditional ergonomic design techniques govern how controls are situated, clearly displayed and intuitive to use, and those approaches can reduce the likelihood of execution errors by the operator, such as manipulating one control when the intention was to adjust a different one. However, without considering the cognitive processes of the operator, ergonomic techniques fall short at preventing errors of intention, such as when a user deliberately adjusts a control, but for the wrong reason. EID aims to minimise the likelihood of that second type of error by incorporating into the interface cognitive factors that take into account how people make decisions and analyse problems.

The following sections describe the EID framework, beginning with two important theoretical concepts used in the approach:

- the abstraction hierarchy;
- the skills, rules and knowledge taxonomy.

Abstraction Hierarchy

The abstraction hierarchy (AH) is a means-end hierarchy that describes the properties of a complex work domain. Each level describes the system in a different way and the breakdown of levels depends on the domain in question. Vicente and Rasmussen propose a five-level hierarchy for process control, such as that which occurs in large-scale industrial plants, although it has the potential to be applied to any control process. The hierarchy is as follows:

- **Functional purpose**: the high-level purpose for which the system was designed (corresponds to the high-level tasks in the CommonKADS Task Model, with the AH levels that follow corresponding to lower-level Task Model activities);
- **Abstract function**: the causal structure of the system, which could involve the flow of mass, energy or information;
- **Generalised function**: a decomposition of sub-functions that enable the high-level functions above;
- **Physical function**: the characteristics of system components and their connections;
- **Physical form**: the physical appearance and location of components.

The means-end relation among levels ensures that moving up the hierarchy reveals the broad goals that the system achieves, while moving down in the hierarchy provides insights into the methods (plans) by which those goals are met. The descriptions of system functions can be characterised in terms of goals. For example, the functional purpose of a nuclear power plant would typically be, "to produce electricity," which is consistent with the terminology of goals. The lower-level functions can be similarly expressed. That goal-based structure has the potential to dovetail with the goal and plan hierarchy of Explicit Models Design (EMD), particularly with respect to higher level goals. It should be noted that proponents of EID have not demonstrated that the abstraction hierarchy forms a complete description of control processes or that its levels are mutually exclusive (e.g., the distinction between abstract and generalised functions), both being important arguments for its use. That is, completeness, consistency and correctness, the hallmarks of good theoretical design, have not been demonstrated.

It is the higher levels of the AH that are not normally addressed in interface design, and which offer the potential to improve system safety by integrating information from lower levels to synthesise a higher-level data representation. For example, in the Three Mile Island (TMI) nuclear power plant accident, operators used a single pressure sensor reading as an indicator of the amount of cooling water in the system. While that was adequate most of the time, under certain boundary conditions, such as those that occurred at the time of the accident, it was not accurate. An EID interface for such a system would reflect the higher levels of the AH by presenting a single indication of the amount of cooling water based on readings from a variety of lower-level sensors. A well-designed ecological interface therefore would display a true (high-level) reading of coolant in the system instead of showing an indirect (low-level) measure inferred from pressure sensors that could not be relied upon under extreme conditions. A more general statement of that notion appears in the "EID Principles" section below.

One caveat is that the EID approach is not entirely prescriptive in that the skill of a designer will influence the resulting interface. Thus, it is conceivable that two different designers could produce substantially different ecological interfaces for a single system. Consequently, it is difficult to claim that all interfaces generated by applying EID in a problem domain will produce adequate data

representations that effectively reduce the likelihood of accidents. Additional long-term studies and industry experience with ecological implementations are needed to demonstrate the usefulness of the approach.

Although the rational arguments for EID are strong, without experimental evidence, it is difficult to argue persuasively that the EID approach would have resulted in an interface that would have prevented the TMI accident. To demonstrate that empirically, it would be necessary to conduct an experiment in which plant designers would apply EID techniques to analyse and redesign a plant like the one at TMI, or significant elements of such a plant. Such an experiment would need to be conducted using designers with no knowledge of that accident, and finding such designers would be hard to imagine. Thus, while it is easy for proponents of EID to state that it would have prevented the accident, providing convincing evidence of that statement would be extremely difficult.

Skills, Rules and Knowledge Taxonomy

Rasmussen (1983) defined three levels of cognitive control describing mechanisms that people use to process information, and those levels constitute the Skills, Rules and Knowledge (SRK) taxonomy.

The lowest level of cognitive control is associated with skill-based behaviour (SBB), which involves automated behavioural patterns, such as those required for riding a bicycle or playing a musical instrument. SBB activities typically require rapid co-ordinated movements that, once mastered, can be performed without conscious effort. Because of parallels with the levels of perceptual control (see the section, "The Role of Perceptual Control Theory"), it could prove fruitful to compare the levels of each with a view to establishing a consistent use of levels in the design and implementation for the UAV/UCAV project.

The second level involves rule-based behaviour (RBB), which is associated with cue-action mappings. Thus, the perception of a certain sign will initiate a particular behaviour according to well-defined rules. Tasks associated with driving a car are considered to be RBB since they involve choosing appropriate actions in response to cues from the environment.

Note that there is not always a well-defined separation between skill-based and rule-based behaviours. Skill-based behaviour, such as playing a musical instrument, is often learned from rules, as in those that correlate the sight of a particular musical note with pressing a specific key on a piano. The transition from rule- to skill-based behaviour makes it difficult to define a division between the two.

The highest level of cognitive control is associated with knowledge-based behaviour (KBB), where problem-solving occurs, typically in conjunction with symbolically represented information. KBB places more intense cognitive demands on the operator but it cannot be avoided, as the performance of complex tasks will nearly always involve some form of KBB.

Rasmussen makes a distinction between the first two levels, which are described as perceptual processing activities, and the third level, which is considered to be an analytical process. That distinction among levels of cognitive control has implications for the design of interfaces. Perceptual processing is typically faster, requires less effort and is less error-prone than analytical processing. Also, it has been demonstrated that operators (particularly those with a high level of skill) tend to favour perceptual behaviour when it is possible. EID encourages the design of interfaces that allow the efficiency of skill-based and rule-based behaviour, while at the same time acknowledging that there will be tasks in complex systems that require analytical processing. Thus, a well-designed interface supports all three modes.

It is important to note that the availability of all three modes of cognitive control in an interface does not guarantee that they will be used appropriately. No matter how much planning goes into the design of a system, the possibility of unforeseen scenarios will continue to exist, and hence the possibility of accidents such as the one at Three Mile Island. Although EID cannot eliminate such accidents entirely, it can help to reduce them substantially if its claims are correct.

EID Principles

The EID approach can be summarised in the statement of three principles, corresponding to the three levels of cognitive control (Vicente and Rasmussen, 1992):

1. *Skill-based behaviour:* To support interaction via time-space signals, the operator should be able to act directly on the display, and the structure of the displayed information should be isomorphic to the part-whole structure of movements (actions).

2. *Rule-based behaviour:* Provide a consistent one-to-one mapping between the work domain constraints and the cues or signs provided by the interface.

3. *Knowledge-based behaviour:* Represent the work domain in the form of an abstraction hierarchy to serve as an externalised mental model that will support knowledge-based problem solving.

The rule for SBB corresponds to principles of direct manipulation. Operators perform best when the interface representation of the system reflects their mental models of process structure and when that representation can be controlled directly, as with a mouse or trackball.

To support RBB, EID advocates providing interface representations for all relevant constraints in the system. That was illustrated in the Three Mile Island example above. In that case, a pressure indicator was used to infer coolant levels without taking into account that that is not a fair assumption under rare circumstances, such as those that occurred at the time of the accident. EID stipulates that each quantity identified in the AH have its own representation in the interface.

The rule that supports KBB requires that the work domain be represented in the interface at all levels of abstraction. That is the result of research (Rasmussen, 1986) that indicates human problem solving can be mapped onto an abstraction hierarchy. When people are faced with a complex problem whose origin cannot be identified quickly, they tend to examine the broader picture of system functions to identify those which are not operating correctly. In doing so, they acquire a wider perspective than if they were to continue searching for problems at a lower level. Descending in the hierarchy from those points allows the operator to focus on the specific component that has failed. That method of problem solving is both efficient and natural to people and therefore an interface representation that corresponds to the abstraction hierarchy is an important element of EID.

It is important to note that there are many practical scenarios in which operators of complex systems would not tend to follow a top-down, abstraction hierarchy approach to problem-solving. Issues surrounding experience, intuition and bias could all influence an operator's tactics. For example, if a system indicated a failure and an operator was aware that a particular part had recently experienced problems, the operator might begin by inspecting that part as the source of the failure, prior to conducting any form of top-down diagnosis. Many other examples of effective local problem-solving can be imagined.

The manner in which a system informs an operator could also affect problem solving. Since the interface offers both high-level and low-level representations of the process, an operator observing an

irregularity in a high-level abstraction of system parameters would naturally proceed by moving down to the lower levels of the hierarchy. In contrast, if an alarm indicates the failure of a particular low-level system, the operator likely would not examine the broad picture of system status immediately, instead choosing to focus on the affected sub-system.

Experimental Support for EID

From a rational perspective, the theoretical arguments behind EID are strong, but, empirically, there is a paucity of experimental data to support the approach.

Several published studies exist that assess the effectiveness of EID in a variety of domains: nuclear power plant simulation (Yamaguchi & Tanabe, 1999), refrigeration plants (Lehane, Toleman & Benecke, 2000) and anaesthesiology (Watson, 1999).

One NASA study has confirmed the benefits of EID in the UAV domain (Dowell, Morphew and Shively, 2003). Operators were observed performing tasks in a traditional interface, an ecological interface and a hybrid. Task performance, situational awareness and workload were improved as a result of the ecological approach.

Such comparisons between existing interfaces and those re-designed using the ecological approach present the possibility that observed improvements are not the result of EID, but instead arise from the process of considering known problems when re-designing the existing interface. Where possible, future studies should investigate re-design using a variety of approaches for comparison with EID.

Yu, Lau, Vicente and Carter (2002) have described a method based on the abstraction hierarchy for quantifying ergonomic performance in an interface. They propose that it could be used in future studies to quantify the effectiveness of experimental interfaces.

Although EID has been around for over a decade, there are relatively few studies like the ones above and the work in ecological interfaces has been confined to a rather small group of researchers. Additional empirical studies by independent researchers showing the benefits of EID would bolster the case for its use.

Ecological Interface Design and the Generic Framework

Given strong rational arguments for the ecological approach to interface design and the empirical support cited here, EID can play a role in the construction of an adaptive interface for UAV/UCAV control. The highly complex nature of UAV/UCAV control tasks and associated high operator workloads reveal the need for optimal interaction among system and user agents, particularly in the context of the stated goal of one operator controlling multiple aircraft. EID's accommodation of both perceptual and analytical cognitive processing serves that goal through the increased efficiency in encouraging skill-based and rule-based behaviours.

Ensuring safe and reliable system operation is of critical importance for UAVs and UCAVs in military operations. For example, the safety of troops on the ground can depend on successful UAV reconnaissance and proper deployment of UCAV ordnance. EID was specifically developed to help ensure the safety and reliability of complex systems.

There are several reasons why EID integrates well with the other, recommended components of the generic framework. One key reason is that the process of designing an ecological interface for the

UAV/UCAV control system would begin with the construction of an abstraction hierarchy, and that corresponds nicely to the hierarchy of goals identified during the creation of the CommonKADS Task Model. The tasks in such an abstraction hierarchy also can be regarded as a high-level subset of the those in the EMD Task Model.

Rasmussen (1998), one of the originators of the EID approach, presents a report on ecological interfaces for UAV/UCAV command and control systems. He recommends the creation of a knowledge base containing information relevant to UAV/UCAV control, such as mission scenarios and rules of engagement, as determined by subject-matter experts. That knowledge-based approach to UAV/UCAV ecological interface design is well suited to the other elements of the generic framework.

Rasmussen's recommendations allow for an interface that supports command personnel in addition to those directly operating the UAVs. Thus, as appropriate, senior personnel can be involved in mission planning and situation assessment in real-time. It is most important that information passed to the higher levels of command be filtered on the basis of relevance, which, in the present case, would be the task of an intelligent system. Suitably filtered knowledge would allow commanders to re-plan UAV/UCAV and other missions within the context of the larger battle plan.

Related to the Rasmussen (1998) work cited above are papers by Flach, Eggleston, Kuperman & Dominguez (1998) and Flach & Kuperman (1998) on Cognitive Systems Engineering (CSE), an approach that involves the creation of an abstraction hierarchy. The first paper contains an analysis of the abstraction hierarchy in the UAV domain that could serve as a model for designers applying the generic framework to an intelligent, adaptive interface. The second paper contains a more general analysis of abstraction hierarchies for military systems that could also apply to UAV/UCAV control.

One caveat on the use of EID is that it does not appear to address implications of automation, such as the boredom and fatigue, that can result from repeated applications of rule-based and skill-based behaviours. Further, it does not address how to detect when operators would become over-reliant on the perceptual behaviours and fail to use knowledge-based skills in situations where they are required.

Forms of Adaptive Aiding

When the help system establishes a need to provide adaptive aiding to users, the control loops in the system will generate behaviour intended to correct the perception that the user requires assistance. The form of assistance will depend on the nature and urgency of the user's needs. The five forms of help described earlier (those associated with tasks being performed completely, consistently, correctly, efficiently and safely) will require different forms of aiding. For example, an unsafe act likely will require a rapid and conspicuous notification to the user, whereas information about user efficiency could be delayed until the demands on the operator have been alleviated.

Assistance could take (but is not limited to) any of the following forms:

- Presentation of simple help messages informing the user of a feature with which the system believes he or she is unfamiliar;
- Presentation of a question asking if the user would like assistance;
- Presentation of a question seeking clarification as to the user's intentions. Such responses can enable the system to discern which of several similar plans the user is currently pursuing;
- The illumination of an indicator (such as a help icon) to notify the user that help is available;
- Brief pop-up descriptions of interface elements when the mouse is over top;
- Offering to complete a task that the user has started;
- An auditory signal to the operator (e.g., a warning tone or a machine-generated spoken message). In situations where the safety of personnel or equipment is at risk, an auditory alarm should accompany the presentation of an on-screen warning to the user;
- Visual cues may be superimposed over the operator's view from aircraft cameras. That would draw upon information contained in the World Model, such as terrain features, enemy positions and targets;
- Haptic feedback could be provided to simulate that which would be felt on a manned aircraft, e.g., control stick shuddering in response to turbulent conditions;
- Tutorials could be offered in the form of post-mission debriefings to provide operators with feedback on performance during the mission. That would provide an opportunity for the system to help the user without disruption during an operation.

Standard documentation is to be available to users on request, in addition to that which is provided automatically. User access to that information will be monitored by the system to formulate User Model beliefs about user knowledge. That documentation may include general information about flight control as well as specific information about the UAV/UCAV interface.

Operator Training

For training new operators, tutorials can be made available that would operate in simulation mode, so as not to endanger personnel or equipment.

- Tutorials should be tailored to the user according to what the software believes the user knows. It may be presented according to a prearranged schedule resulting from an agreement (a "contract" in PACT terminology) between the user and system (e.g., a daily mini-tutorial on a feature with which the user is not familiar);

- Interactive tutorial: Some tutorials could be interactive, whereby the system would step through the description of the procedure while the user carried it out;

- In addition to providing instruction about features the system believes the user does not know about, tutorials could provide guidance on how to solve an application-specific problem a user may have. That might occur in response to the user asking, "How do I finish the task that I have begun?";

- Tutorials offered in the form of post-mission debriefings (see the previous section) could also provide operators with feedback following simulated missions.

Design and Implementation within the Generic Framework

The current project examined a variety of theoretical approaches to construct a comprehensive, integrated framework for the design and implementation of an intelligent, adaptive, agent-based system for single-operator control of multiple UAVs/UCAVs. The resulting **generic framework** is composed of elements from the following design approaches:

- **CommonKADS (CK)** – a knowledge management and engineering methodology that guides the systematic analysis and design of intelligent systems;
- **MAS-CommonKADS (MAS-CK)** – a modification of the CommonKADS methodology that adds an infrastructure for multiple agent development;
- **IDEF Standards** – a complement to the CommonKADS methodology through its more effective support for temporal modelling and ontology construction;
- **Explicit Models Design (EMD)** – a methodology for building models that identify and compartmentalise the knowledge required by intelligent systems;
- **Perceptual Control Theory (PCT)** – a feedback control system model for goal-directed behaviour in a system;
- **Ecological Interface Design (EID)** – a set of techniques for constructing safe and reliable user interfaces based on levels of cognitive control.

Overviews of the individual approaches were given, and common and complementary elements of the framework have been highlighted in the discussion. The integration of the above techniques into a comprehensive, cross-disciplinary design approach should serve goals of reducing operator workloads and manning requirements, while generating a robust, maintainable and reliable UAV/UCAV system.

Following is a description of the recommended procedure for designing and implementing a knowledge system within the generic framework. Steps are designed to be pursued in the sequence presented.

To facilitate the presentation of the procedure, a legend is provided of the models that comprise the CommonKADS (CK) knowledge and engineering methodology and those used in Explicit Models Design (EMD).

CommonKADS (CK):

- Organisation Model;
- Task Model (CK);
- Agent Model;
- Knowledge Model;
- Communication Model;
- Co-ordination Model (MAS-CommonKADS);
- Design Model.

Explicit Models Design (EMD):

- Task Model (EMD);
- User Model;
- System Model;
- Dialogue Model;
- World Model.

Proposed Generic Framework

- Construct the Organisation Model to describe the command and control structure within which the project will be developed;
- Construct the Task Model (CK), including task hierarchies for all agents identified above (use IDEF3 to represent the hierarchies);
- Construct the Agent Model identifying all user and system agents and their relationships;
- Adapt the Task Model (CK) to a five level Abstraction Hierarchy according to the levels specified by the EID approach;
- Generate the Task Model (EMD) by extending the Task Model (CK) to produce task hierarchies for all agents using PCT-based hierarchical goal analysis;
- Develop the User Model according to the need to track user preferences and knowledge;
- Specify the content of the System Model to enable representation and use of system preferences and knowledge;
- Design the World Model to contain required information about the environment necessary for the knowledge system to operate effectively;
- Specify the Dialogue Model, Communication Model and Co-ordination Model to govern the format and content of communication among agents (ensure that the ability exists for agents to provide feedback to one another);
- Use IDEF5 to design an ontology to represent the contents of all Explicit Models;
- Develop the Knowledge Model to encapsulate the ontology and an associated knowledge base containing information from all Explicit Models;
- Within the Knowledge Model, represent the Task Model (EMD) as a hierarchy of PCT loops that use plan recognition and plan generation to form input perceptions and output behaviours;
- Create the Design Model to produce design specifications for the target knowledge-based system;
- Apply EID techniques to develop specifications for the user interface to the knowledge system.

References

Akkermans, H., Gustavsson, R. & Ygge, F. (1998). An integrated structured analysis approach to intelligent agent communication. <http://www.enersearch.se/knowledgebase/publications/conferencejournals/IFIP98/ifip98.pdf>.

Anderson, J. R. (1993). *Rules of the Mind*. Hillsdale, NJ: Erlbaum.

Arenas, A.E. & Barrera-Sanabria, G. (2002). Applying the MAS-CommonKADS methodology to the flights reservation problem: Integrating coordination and expertise. <http://lpt.uni-mb.si/jckbse2002/aplication/Apl_Login/upcamera/36-jckbse.pdf>.

Card, S. K., Moran, T. P. & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Erlbaum.

CFEC (2002). Canadian Forces joint concept development and experimentation plan (2002). Plan Pegasus, Canadian Forces Experimentation Centre (CFEC).

Cognitive engineering laboratory: Ecological interface design. <http://www.mie.utoronto.ca/labs/cel/research/eid.html>.

Dixon, S.R. and Wickens, C.D. (2003). Control of multiple-UAVs: A workload analysis.

Dixon, S.R., Wickens, C.D. and Chang, D. (2003). Comparing quantitative model predictions to experimental data in multiple-UAV flight control.

Dowell, S., Morphew, E. & Shively, J. (2003). An ecological approach to the design of UAV ground control station (GCS) status displays. *AUV SI*, Jul. 15-17, 2003, Baltimore, MD, USA.

Ecological interface design. <http://individual.utoronto.ca/olivier/interest/eid.html>.

Edwards, J. L. (1990). Proposed architecture for SDBMS–2. Report prepared for the Defence and Civil Institute of Environmental Medicine (DCIEM) by Artificial Intelligence Management and Development Corporation. Toronto, Ontario, Canada.

Edwards, J. L. (1994). Distributed artificial intelligence and knowledge-based systems. In H. U. Steusloff (Ed.), Distributed systems modelling emphasizing object-orientation. Technical Report AC/243 (Panel 11) TR/s. North Atlantic Treaty Organization Defence Research Group. NATO Unclassified.

Edwards, J. L. & Hendy, K.C. (2000). A testbed for intelligent aiding in adaptive interfaces. Paper presented at the Spring Symposium on Adaptive User Interfaces. Stanford University, March 20-22, 2000. Technical Report SS-00-01, AAAI Press.

Edwards, J. L., Scott, G. & Hendy, K.C. (2003). A testbed for adaptive aiding using the LOCATE workspace layout tool. (In preparation)

Flach, J.M., Eggleston, R., Kuperman, G. & Dominguez, C. (1998). SEAD and the UCAV: A preliminary cognitive systems analysis. USAF Research Laboratory Report AFRL-HE-WP-TR-1998-0013.

Flach, J.M. & Kuperman, G. (1998). Victory by design: War, information and cognitive systems engineering. USAF Research Laboratory Report AFRL-HE-WP-TR-1998-0074.

Franklin, S. & Graesser, A. (1996) "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents," Intelligent Agents III, Berlin: Springer Verlag, 21-35.

Hendy, K. C., Beevis, D., Lichacz, F., and Edwards, J. L. (2002). Analyzing the cognitive system from a perceptual control theory point of view. In, Cognitive systems engineering in military aviation environments: Avoiding cogminutia fragmentosa!: A report produced under the auspices of The Technical Cooperation Programme Technical Panel HUM TP-7 Human Factors in Aircraft Environments. Human Systems IAC, Wright-Patterson AFB, OH.

Henesey, L.E., Notteboom, T.E. & Davidsson, P. (2003). Agent-based simulation of stakeholders relations: An approach to sustainable port terminal management. <http://www.ide.hk-r.se/~pdv/Papers/IAME2003.pdf>.

Hou, M. (2003) A Conceptual Framework for Optimizing Operator-agent Interaction. DRDC Toronto Technical Report, TR 2004-059.

IEEE-1220 (1994). Application and management of the systems engineering process. Piscataway, NJ, USA: Institution of Electrical and Electronic Engineers.

Iglesias, C.A., Garijo M., González, J.C. & Velasco, J.R. (1996). A Methodological Proposal for Multiagent Systems Development Extending CommonKADS. <http://citeseer.nj.nec.com/rd/86431430%2C80375%2C1%2C0.25%2CDownload/http://citeseer.nj.nec.com/cache/papers/cs/4919/http:zSzzSzwww.gsi.dit.upm.eszSz%7EmastzSzpszSzkaw96.pdf/iglesias96methodological.pdf>.

Iglesias, C.A., Garijo M., González, J.C. & Velasco, J.R. (1996). A Methodological Proposal for Multiagent Systems Development Extending CommonKADS. <http://citeseer.nj.nec.com/rd/86431430%2C80375%2C1%2C0.25%2CDownload/http://citeseer.nj.nec.com/cache/papers/cs/4919/http:zSzzSzwww.gsi.dit.upm.eszSz%7EmastzSzpszSzkaw96.pdf/iglesias96methodological.pdf>.

Iglesias, C., Garijo, M., Gonzalez, J. C. & Velasco, J R. (1998). Analysis and design of multiagent systems using MAS-CommonKADS, *Intelligent Agents IV*, Springer-Verlag, 313-326.

Iglesias, C.A., Garijo, M. & González, J.C. (1999). A survey of agent-oriented methodologies. <http://citeseer.nj.nec.com/cache/papers/cs/994/http:zSzzSzwww.gsi.dit.upm.eszSz%7EmastzSzpszSzatal98.pdf/iglesias99survey.pdf>.

Iglesias, C., Gonzalez, J.C. & Velasco, J.R. (1998). A Fuzzy-neural multiagent system for optimisation of a roll-mill application, *Proceedings of the 11th int. conference on industrial and engineering applications of artificial intelligence and expert systems, Benicasim, Spain*, 596-605.

Jameson, S.M. (2001). Architectures for distributed information fusion to support situation awareness on the digital battlefield.

John, B.E. & Kieras, D.E. (1996a). The GOMS family of user interface analysis techniques: Comparison and contrast. <ftp://www.eecs.umich.edu/people/kieras/GOMS/Compare-GOMS.pdf>.

John, B.E. & Kieras, D.E. (1996b). Using GOMS for user interface design and evaluation: Which technique? <ftp://www.eecs.umich.edu/people/kieras/GOMS/Which-GOMS.pdf>.

Jones, D.M., Bench-Capon, T.J.M. & Visser, P.R.S. (1998). "Methodologies for Ontology Development", in *Proc. IT&KNOWS Conference, XV* IFIP World Computer Congress, Budapest.

Kennaway, R. (2000). Archy: A multi-legged robot. <http://www2.cmp.uea.ac.uk/~jrk/PCT/Archy/Archy.html>.

Knowledge Based Systems, Inc. (1994). Information integration for concurrent engineering (IICE) IDEF5 method report. Armstrong Laboratory, Wright-Patterson Air Force Base, Ohio.

Knowledge Based Systems, Inc. (1995). Information integration for concurrent engineering (IICE) IDEF4 object-oriented design method report. Air Force Systems Command, Wright-Patterson Air Force Base, Ohio.

Lehane, P., Toleman, M. & Benecke, (2000). Applying ecological interface design to experimental apparatus used to monitor a refrigeration plant. First Australasian User Interface Conference, Jan. 31 – Feb. 3, 2000, Canberra, Australia.

Lesh, N.B., Rich, C. & Sidner, C.L. (1999). "Using plan recognition in human-computer collaboration", *Proceedings of the international conference on user modeling, June 1999*, 23-32.

Lydiard, T. J. (1995). Using IDEF3 to capture the air campaign planning process. University of Edinburgh, Scotland.

Mayer, R. J. (ed.) (1992). IDEF1 information modeling. Knowledge Based Systems, Inc., College Station, Texas.

Mayer, R. J., Menzel, C. P., Painter, M. K., deWitte, P. S., Blinn, T. & Perakath, B. (1995). Information integration for concurrent engineering (IICE) IDEF3 process description capture method report. Knowledge Based Systems, Inc., College Station, Texas.

National Institute of Standards and Technology (1993a). Integration definition for function modeling (IDEF0). National Technical Information Service, Springfield, VA.

National Institute of Standards and Technology (1993b). Integration definition for information modeling (IDEF1X). National Technical Information Service, Springfield, VA.

Nwana, H. (1996). Software agents: An overview. *Knowledge Engineering Review, 11(3)*, 1-40.

Powers, W. T. (1973). *Behavior: The control of perception*. New York: Aldine De Gruyter.

Powers, W. T. (1989). An outline of control theory. In, R. S. Marken (ed.), *Living control systems — selected papers of William T. Powers*. Gravel Switch, KY: The Control Systems Group, Inc., 253-293.

Powers, W. T. (1990a). A hierarchy of control. In, R. J. Robertson and W. T. Powers (Eds.), *Introduction to modern psychology: The control-theory view*. Gravel Switch, KY: The Control Systems Group Inc., 59-82.

Powers, W. T. (1990b). How Behavior Becomes Organized. In, R. J. Robertson and W. T. Powers (Eds.), *Introduction to modern psychology: The control-theory view*. Gravel Switch, KY: The Control Systems Group Inc., 95-108.

Rasmussen, J. (1983). "Skills, rules, knowledge: Signals, signs, and symbols and other distinctions in human performance models," *IEEE TransActions on Systems, Man, and Cybernetics, Vol. SMC-13*, 257-267.

Rasmussen, J. (1986). *Information processing and human-machine interaction: An approach to cognitive engineering*. New York: Elsevier Science (North-Holland).

Rasmussen, J. (1998). Ecological interface design for complex systems: An example: SEAD UAV systems. USAF Research Laboratory Report AFRL-HE-WP-TR-1999-0011.

Rich, C., Sidner, C.L. & Lesh, N.B. (2001). "COLLAGEN: Applying collaborative discourse theory to human-computer interaction", *Artificial intelligence magazine, 22(4)*, 15-25.

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W. & Wielinga, W. (2000). *Knowledge engineering and management: The CommonKADS methodology*. MIT Press: Cambridge, Massachusetts.

Sommerville, I. (1995). *Software engineering*. Harlow, UK: Addison-Wesley.

Taylor, R.M. (2001). Cognitive Cockpit systems engineering: Pilot authorisation and control of tasks. Defence Science and Technology Laboratory, Human Sciences, A2 Bldg, Rm. G007, Ively Gate, Ively Road, Farnborough, Hants GU14 OLX, UK.

Vicente, K.J. (1999). *Cognitive work analysis: Toward safe, productive, and healthy computer-based work*.. Mahwah, NJ : Lawrence Erlbaum Associates.

Vicente, K.J. & Rasmussen, J. (1992). Ecological interface design: theoretical foundations. *IEEE transactions on systems, man and cybernetics, SMC-22*, 589-606.

Watson, M., Russell, W.J. & Sanderson, P. (1999). Ecological interface design for anaesthesia monitoring. *Proceedings of the 9th Australasian Conference on Computer-Human Interaction* OzCHI99.Wagga Wagga, Australia: IEEE Computer Society Press, 78-84.

Wooldridge, M.J. & Jennings, N.R. (eds.) (1995a). *Intelligent agents, lecture notes in artificial intelligence*, Springer Verlag.

Wooldridge, M. & Jennings, N.R. (1995b). Intelligent agents: Theory and practice, *The knowledge engineering review, 10(2)*, 115-152.

Yamaguchi, Y. & Tanabe, F. (2002). Creation and evaluation of an ecological interface system for operation of nuclear reactor system, Enlarged Halden Programme Group Meeting, Gol, Norway, September 8-13, 2002.

Yu, X., Lau, E., Vicente, K.J. & Carter, M.W. (2002). Toward theory-driven, quantitative performance measurement in ergonomics science: The abstraction hierarchy as a framework for data analysis. *Theoreticl Issues in Ergonomic Science, 3(2)*, 124-142.

List of abbreviations/acronyms

AH	Abstraction Hierarchy
AI	Artificial Intelligence
CK	CommonKADS
COGPIT	Cognitive Cockpit
CSE	Cognitive Systems Engineering
DERA	Defence Evaluation and Research Agency
DRDC	Defence R & D Canada
EID	Ecological Interface Design
EMD	Explicit Models Design
GIS	Geographical Information System
HCI	Human-Computer Interaction
HGA	Hierarchical Goal Analysis
IAI	Intelligent, Adaptive Interface
ICAM	Integrated Computer-Aided Manufacturing
IDEF	ICAM Definition
IISR	Integrated Intelligence, Surveillance and Reconnaissance
JESS	Java Expert System Shell
KADS	Knowledge Acquisition and Design Structuring
KBB	Knowledge-Based Behaviour
KBS	Knowledge-Based Systems
MAS-CK	Multi-Agent System CommonKADS
MAS-CommonKADS	Multi-Agent System CommonKADS
MFTA	Mission, Function, Task Analysis
MVC	Model-View-Controller
NASA	National Aeronautics and Space Administration
OCL	Object Constraint Language
O-O	Object-Oriented
PACT	Pilot Authorisation and Control of Tasks
PCT	Perceptual Control Theory
RBB	Rule-Based Behaviour
SBB	Skill-Based Behaviour
SEAD	Suppression of Enemy Air Defences
SRK	Skills, Rules and Knowledge
TMI	Three Mile Island
UAV	Unmanned Air Vehicle
UCAV	Unmanned Combat Air Vehicle
UML	Unified Modelling Language
USAF	United States Air Force

DOCUMENT CONTROL DATA SHEET

1a. PERFORMING AGENCY

Artificial Intelligence Management and Development Corporation, 206 Keewatin Avenue, Toronto, ON M4P 1Z8 CANADA

2. SECURITY CLASSIFICATION

UNCLASSIFIED
- Unlimited distribution -

1b. PUBLISHING AGENCY

DRDC Toronto

3. TITLE

(U) A Generic, Agent-Based Framework for Design and Development of UAV/UCAV Control Systems

4. AUTHORS

Jack L. Edwards

5. DATE OF PUBLICATION

February 27 , 2004

6. NO. OF PAGES

59

7. DESCRIPTIVE NOTES

8. SPONSORSHIP/MONITORING/CONTRACTING/ASKING AGENCY

Sponsoring Agency:
Monitoring Agency:
Contracting Agency: DRDC Toronto
Tasking Agency:

9. ORIGINATOR'S DOCUMENT NO.	10. CONTRACT, GRANT AND/OR PROJECT NO.	11. OTHER DOCUMENT NOS.
Contract Report CR 2004-062	PWGSC W7711-037857/001/TOR	AC261

12. DOCUMENT RELEASABILITY

Unlimited announcement

13. DOCUMENT ANNOUNCEMENT

Unlimited announcement

14. ABSTRACT

Unmanned Air Vehicles (UAVs) and Unmanned Combat Air Vehicles (UCAVs) are being investigated for use as a new Integrated Intelligence, Surveillance, and Reconnaissance (IISR) platform within the Canadian Forces. At the moment UAV/UCAV control is operator intensive and can involve high levels of workload. In an effort to alleviate those conditions and reduce manning requirements, the current project examined a variety of theoretical approaches to construct a comprehensive, integrated approach to the design and implementation of an intelligent, adaptive, agent-based system for UAV/UCAV control. The resulting generic framework was constructed from the elements of the following design approaches: CommonKADS, MAS-CommonKADS, IDEF Standards, Explicit Models Design, Perceptual Control Theory and Ecological Interface Design. This report provides overviews of each of those approaches and highlights common and complementary elements as part of a recommended generic framework. A sequence for applying the generic framework is provided. The proposed integration of the above techniques into a comprehensive, cross-disciplinary design approach will help serve the goals of reducing operator workloads and manning requirements, while generating a robust, maintainable and reliable system.

RÉSUMÉ

Unmanned Air Vehicles (UAVs) and Unmanned Combat Air Vehicles (UCAVs) are being investigated for use as a new Integrated Intelligence, Surveillance, and Reconnaissance (IISR) platform within the Canadian Forces. At the moment UAV/UCAV control is operator intensive and can involve high levels of workload. In an effort to alleviate those conditions and reduce manning requirements, the current project examined a variety of theoretical approaches to construct a comprehensive, integrated approach to the design and implementation of an intelligent, adaptive, agent-based system for UAV/UCAV control. The resulting generic framework was constructed from the elements of the following design approaches: CommonKADS, MAS-CommonKADS, IDEF Standards, Explicit Models Design, Perceptual Control Theory and Ecological Interface Design. This report provides overviews of each of those approaches and highlights common and complementary elements as part of a recommended generic framework. A sequence for applying the generic framework is provided. The proposed integration of the above techniques into a comprehensive, cross-disciplinary design approach will help serve the goals of reducing operator workloads and manning requirements, while generating a robust, maintainable and reliable system.

15. KEYWORDS, DESCRIPTORS OR IDENTIFIERS

(U) ADAPTIVE INTERFACE; AGENT MODEL; AIRBORNE PLATFORMS; COMMONKADS; ECOLOGICAL INTERFACE DESIGN; EXPLICIT MODELS DESIGN; EMD; GENERIC, AGENT-BASED FRAMEWORK; HUMAN-COMPUTER INTERACTION; IDEF; INTEGRATED INTELLIGENCE, SURVEILLANCE, AND RECONNAISSANCE PLATFORM; IISR; INTELLIGENT INTERFACE; KNOWLEDGE MODEL; KNOWLEDGE-BASED SYSTEMS; KBS; MAS-COMMONKADS; PERCEPTUAL CONTROL THEORY; PCT; REMOTE OPERATION; TASK MODEL; UAV; UCAV; UNMANNED AIR VEHICLES; UNMANNED COMBAT AIR VEHICLES